university of groningen / Faculty of Science and Engineering / Biomedical Engineering

Exploring and comparing data selection methods in the pre-processing step of a deep learning framework for automatic tumor segmentation on PET-CT images

Ekaterina Korotina

S-4880641

Department of Radiation Oncology/ Data Science Center in Health (DASH), UMCG

Period: 07/02/2022 - 22/07/2022

Master's project

1st Examiner: dr. ir. P.M.A. (Peter) van Ooijen, Scientific Researcher / Associate Professor, Faculty of Medical Sciences, RUG

2nd Examiner: dr. ir. K. (Kelvin) Ng Wei Siang, Research Physicist / Medical Physicist, Faculty of Medical Sciences, RUG

Daily Supervisor: Alessia de Biase, PhD candidate in Deep Learning for Radiotherapy, Department of Radiation Oncology, UMCG

Master Thesis title:

Exploring and comparing data selection methods in the pre-processing step of a deep learning framework for automatic tumor segmentation on PET-CT images

Abstract:

Automatic segmentation of primary tumors in oropharyngeal cancer patients using PET/CT images and deep learning has the potential to improve radiation oncology workflows. However, 2D tumor segmentation using deep learning is a data imbalance problem and a method of PET and CT slice selection affects the convergence of the deep learning model. The aim of the current project was to find a way to select sequences to improve the performance of the existing deep learning segmentation model. To select the 'right amount' of sequences without tumor in an unsupervised manner, clustering methods were explored. The trained clustering algorithms were used to group the training and validation data of the existing segmentation model in into clusters. The performance of the proposed method was assessed using the existing segmentation model. The promising results of the proposed data selection method were confirmed by improved metrics of the segmentation model (mean dice score coefficient, precision and recall).

Key words:

Data selection, class imbalance, PET imaging, features extraction, undersampling clustering method

Table of Contents

List of symbols and abbreviations

PET – Positron Emission Tomography

CT – Computer Tomography

MRI – Magnetic Resonance Imaging

UMCG – University Medical Center Groningen

OCC – One-class classification

SUV – Standardized Uptake Value

FDG – Fluorodeoxyglucose

ROI – Region of Interest

GTV – Gross Tumor Volume

MaxSUV – Maximum Standardized Uptake Value

MinSUV– Minimum Standardized Uptake Value

MeanSUV – Mean Standardized Uptake Value

Std – Standard deviation

WCSS – Within-Cluster Sum-of-Squares

FCM – Fuzzy C-means

DSC – Dice Score Coefficient

1 Introduction

1.1 Background

Oropharyngeal cancer is one of the most common types of cancer around the world. Prior to the treatment, usually radiologists manually segment the tumor and the organs at risk (see Figure 1) on images such as CT, PET, and MRI. However, the surrounding normal tissue is very similar to the tumor in the head and neck area, which makes it more difficult to segment. Moreover, it is a very time-consuming task because the radiologists have to perform contouring in all three perpendicular cross-sections (axial, sagittal, coronal planes). Hence the automatic segmentation of primary tumors in oropharyngeal cancer is of great interest since it has the potential to improve radiation oncology workflow [1].

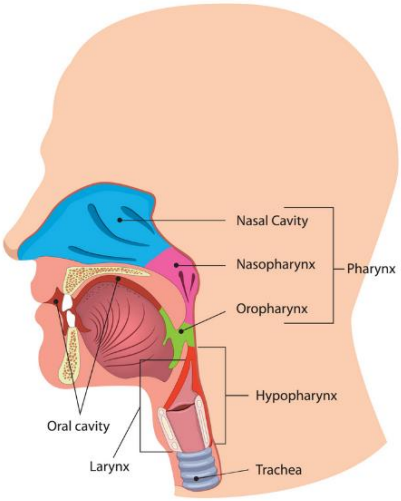

Figure 1. Organs at risk for oropharyngeal cancer [2]

Therefore, at the Department of Radiation Oncology of the University Medical Center Groningen (UMCG) a project was started for the development of a 2D deep learning model for automatic tumor segmentation on PET and CT images [3]. The proposed model trains on 2D slices extracted from PET and CT images of oropharyngeal cancer patients from the UMCG,

which are processed in the form of sequences. A sequence consists of three consecutive slices of concatenated PET-CT images. The 'golden standard' of tumor contours, used in the model, are the manual tumor segmentations provided by the radiation oncologists. They are used as binary masks where pixel values of 1 correspond to tumor and 0 to not tumor. The original 3D volume of each image modality has a dimension of 144x144x144 pixels. Therefore, each sequence used for training is 432x144 pixels.

Tumor segmentation is, as many medical imaging problems, a class imbalanced problem: in a 3D volume, far more slices not containing tumor (negative slices) are present than the ones containing tumor (positive slices). If all slices are considered for training, then the network will exhibit bias towards the majority (negative) class, so an appropriate selection is necessary.

In addition, the area of the head and neck on PET images is also challenging for the model when trying to identify a tumor because of the presence of lymph nodes and the brain with similar brightness which gives rise to a high number of false positive results.

In the paper [3] the segmentation model was trained on positive sequences (95% of the total training data)chosen by adding a constraint on the minimum percentage of tumor pixels present; and negative sequences (5%) which were randomly selected.

The current project explores alternative ways to select negative sequences, dealing with the class imbalanced problem, using PET images information.

1.2 Literature overview

The field of slice selection to solve the class imbalance problem for 2D automatic tumor segmentation has been surprisingly unattended until recently, as the majority of the literature on tumor segmentation performed

slice selection setting a certain threshold value on pixels that corresponds to tumors or did not talk about the pre-processing step at all. In this section different methods for handling class imbalance problems are discussed.

Class imbalance is one of the biggest challenges of the machine learning field. Imbalanced data sets degrade the performance of data mining and machine learning techniques as the overall accuracy and decision making are biased towards the majority class, which leads to misclassifying the minority class samples [4]. Hence, the class imbalance issue is an important topic for the researchers to tackle.

Methods for handling class imbalance in machine learning can be grouped into three categories: data-level techniques, algorithm-level methods, and hybrid techniques [5].

In the data-level approaches the most common ones are oversampling and undersampling techniques. Authors of [12] performed classification of diabetic neuropathy according to MRI images. They applied 73 different oversampling techniques to the dataset in order to deal with class imbalance. Another representative example is [13]. In this paper the authors augmented the training examples based on the ratios of imbalanced classes to solve the imbalanced class problem for deep learning based breast cancer histopathological image classification. One-class classification (OCC) learning algorithms are also known as recognition-based undersampling methods dealing with class imbalance, which work by modeling the classifier on the representation of the minority class. In paper [14] the authors used neural networks and proceeded to learn only from the examples of minority class rather than trying to recognize the different patterns from examples of majority class and minority class. In [15] the authors described the OCC method. They improved the method by adding a calculation of image complexity and tested on four imbalanced datasets of medical images. In addition to the OCC method, also the cluster-based undersampling method

showed its potential. Authors in [16] used a cluster technique that aims to group objects that have similar characters into the same cluster. In the cluster method the first stage was to determine the best number of clusters (indicated with K). The cluster center of these cluster groups was used as a representation of all data that was used as a sample of the majority of classes. Samples in each cluster group were taken randomly and the remaining data was eliminated (undersampling). Furthermore, the majority class sample was combined with the minority class sample to form a new balanced training dataset. In [17] a cluster-based undersampling method was used to handle the data imbalance problem in breast cancer classification. K-means clustering algorithm and Boosted C5.0 were used to select the datasamples located near the tumor boundary. To evaluate the proposed classifier the performance of the proposed model was compared to the baseline approaches. The method from this article performed with less time and better results on statistical parameters as speceficity and sensitivity.

There are several works dealing with the class imbalance problem using algorithm-level approaches [6,7,8,9]. In [6], for instance, authors proposed a new learning rule for Spiking neural networks (SNNs) to solve the medical image class imbalance problem. The rule is called an imbalanced reward-modulation spike-timing-dependent plasticity (R-STDP). They used an imbalanced reward coefficient for the R-STDP learning rule to set the reward from the minority class to be higher than that of the majority class, and this reward coefficient helped to set the class-dependent rewards according to the data statistic of the training dataset. The authors of [7] designed a new recurrent generative adversarial architecture called RNN-GAN to handle imbalance data problem in cardiac MRI image segmentation.They used mixed adversarial loss and categorical accuracy loss in their novel model. The proposed model improved semantic segmentation accuracy on the MRI images dataset. The performance of the model was assessed using

the dice score coefficient. In [8] the authors discussed a new conditional generative refinement network with three components: a generative, a discriminative, and a refinement network. The proposed method was able to mitigate the problem of data imbalance through ensemble learning for simultaneous liver and lesion segmentation, microscopic cell segmentation and brain tumor segmentation. The outcome was compared to the recent popular state-of-the-art methods such as patient-wise mini-batch normalization. To evaluate the proposed method, authors trained their network on the liver tumor segmentation dataset and compared the results according to the dice score. Authors of [9] described novel algorithm-level technique based on the Offset Curves (OsC) loss. The OsC loss consists of three main fitting conditions: pixel-level segmentation, area around the boundaries (offset curves), and length of the offset curves. The loss was used to train the brain tumor segmentation and the vessel extraction models.The outcome of the designed model was evaluated with the help of dice score coefficient and other metrics and for the brain tumor segmentation the dice score coefficient was higher than for other state-of-the-art models using for example FCN architecture.

A hybrid approach is also widely used for handling class imbalance. In the paper [10], authors combined data-level and algorithm-level approaches in order to train the model for a deep learning oral cancer classifier. They augmented the training samples based on the ratios of imbalanced classes to oversample the dataset, then they randomly undersampled the dataset and trained the model using Convolutional Neural Network (CNN) classifier and Cross-Entropy (CE) loss metric as an algorithm-level approach. In the paper [11] the authors used a hybrid method for handling class imbalance of skin-disease classification. This method consists of the data level method of balanced mini-batch logic and a real-

time image augmentation using the algorithm-level approach to design a new loss function.

1.3 Aim and objectives

The aim of the current project was to find a way to select sequences in order to improve the performance of the deep learning segmentation model of [3], which means more accurate automatic tumor segmentation. Since the model showed to be highly affected by PET intensity values, the first goal of this study will be using this image modality to perform undersampling clustering. To select the "right amount" of sequences without tumor in an unsupervised manner clustering methods were explored. Secondly, the different sequence selection methods were compared and discussed based on the improvement of the performance of the segmentation model. To achieve the aim of the project, the following objectives were identified:

- Identify different relevant features describing PET images.
- Analyze and compare clustering methods for data selection using only PET imaging.
- Apply chosen methods of sequence selection to the training and the validation data of the segmentation model.
- Discuss the differences in performance of the segmentation model when different sequence selection methods are used.

Figure 1 shows the workflow planned to implement in the current project.

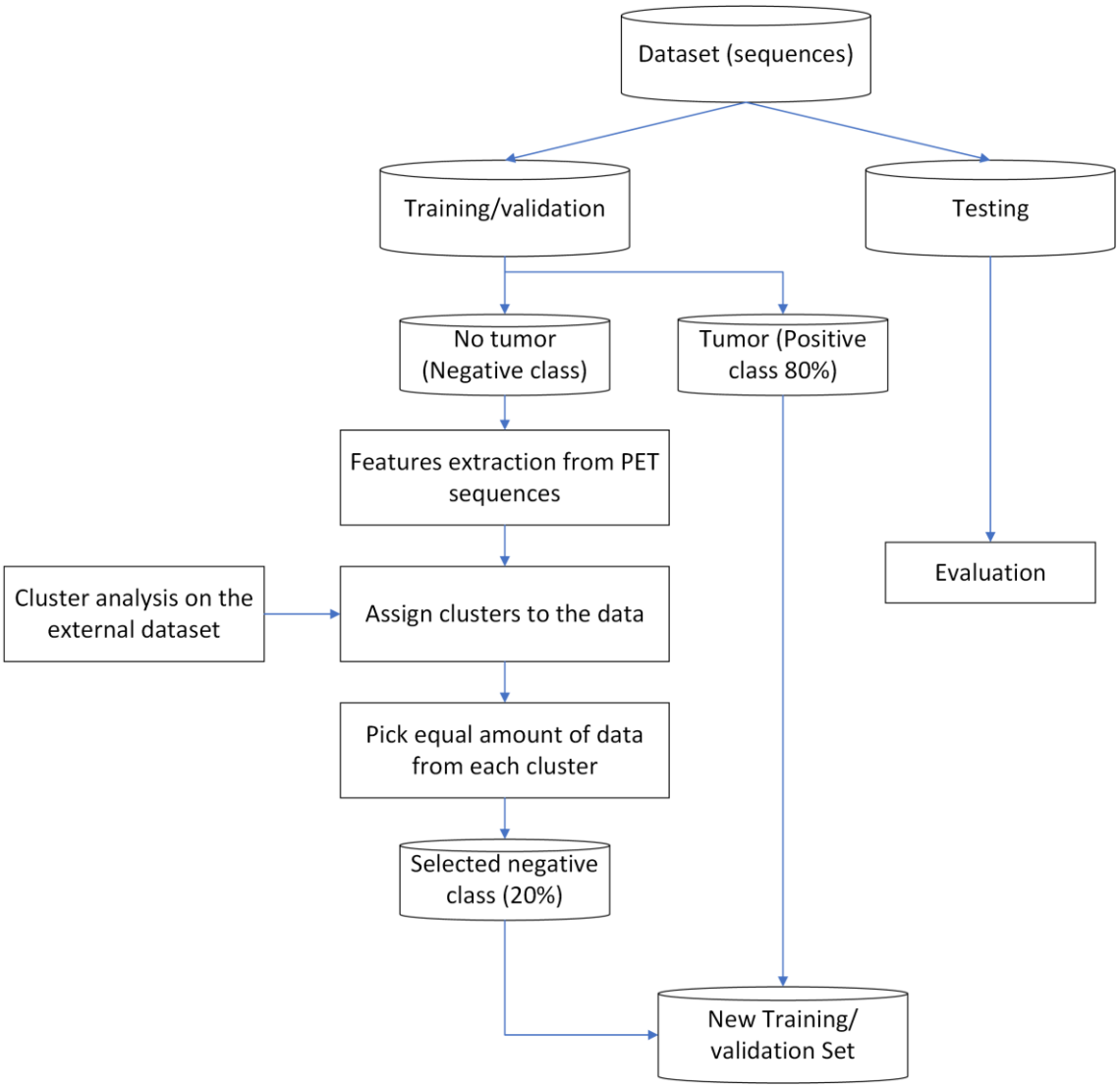

Figure 2. Undersampling clustering-based method used in current project. Image adopted from [16].

2 Materials and Methods

The method used to achieve the aim of the thesis is cluster analysis. In the first part of this chapter, the data used for this project is described. In the second part, the different steps to perform cluster analysis are described, clustering techniques and sequence selection method are discussed.

2.1 Data

2.1.1 PET images. Clinical information

Positron emission tomography (PET) is a procedure in nuclear medicine that measures metabolic activity of the cells in different body tissues. PET is a combination of nuclear medicine and biochemical analysis [18].

PET imaging works by detecting two high-energy photons that coincide in time and are released by a radioisotope that emits positrons. PET imaging has unique characteristics for both very high sensitivity and precise determination of the in vivo concentration of the radiotracer due to the physics of the emission and the detection of the coinciding photons. PET imaging has become a popular clinical modality for oncology, cardiovascular, and neurological applications. PET scan radionuclides are created by attaching a radioactive atom to chemical substances that are naturally used by the organ or tissue during its metabolic process [19].

PET scans produce exact, three-dimensional images of the interior of the human body. The images can clearly show the part of the body being investigated, including any abnormal areas, and can highlight how well certain areas of the body are functioning. A PET scan is a useful tool for diagnosing a number of diseases, including cancer. Head and neck cancer imaging is especially necessary for radiotherapy treatment planning. PET is

used because of its ability to assess tumor metabolic status. PET has outperformed CT and MRI in diagnosing and distinguishing recurrence from post-radiation effects and surgical scars in head and neck tumor sites. In addition, PET is superior to CT and MRI in the detection of cervical lymph node status in cases of head and neck cancer. However, PET is limited by a lack of anatomical markers and it is difficult to find suspicious findings precisely because of the low amount of background tracer absorption [20].

2.1.2 PET images. Imaging information

To visualize PET scans one of the three perpendicular cross-section (axial, sagittal, coronal planes) is usually selected. Primary tumor regions present pixels with higher intensity compared to non cancerous tissues, however the metastatic lymph nodes and the brain show similar brightness as the tumor. Figure 3. shows an example of PET slice from one of the perpendicular cross-sections with bright pixels due to the brain activity. This causes difficulties in automatic tumor detection.

A simple way to determine activity in PET imaging is the standardized uptake value (SUV). SUV measures the relative uptake in a region of interest. Its calculation depends on a precise knowledge of the injected dose quantity and time [21]. The intensity of normal tissues should be within the lower-to-middle portion of the dynamic range while the upper range are used to demonstrate the range of intensities that might exist in pathological processes characterized by high glycolytic activity [22].

SUV is a dimensionless ratio that has traditionally been used by nuclear medicine professionals to differentiate between "normal" and "abnormal" levels of uptake. It is a semi-quantitative parameter defined as the ratio of activity per unit volume of a region of interest (ROI) to activity per unit whole body volume. It was intended to be a simplified method of quantifying uptake rather than true quantification via compartmental and

kinetic modeling. A SUV equal to 2.5 and higher is generally considered to be indicative of the malignant tissue or the tumor. However, there has been a wide range of SUV values reported for different similar diseases. It is important to say that a SUV around 2.5 can be measured in non-malignant regions. In other words, small tumors can also exhibit a maximum SUV of less than 2.5. The SUV was created to determine whether a region may be considered as ‘tumor‘ or ‘malignant‘ but may have limitation for determining the contours of a tumor [23].

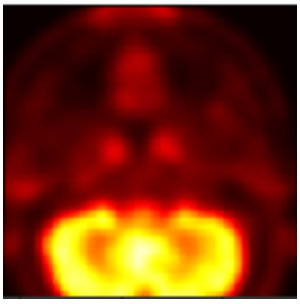

Figure.3. An example of PET slice with clear representation of the brain activity.

2.1.3 Data description

The data used for the current project is the same as in [3]. Imaging data was collected at the UMCG. For this project a total of 114 patients were used. PET, CT and GTV (gross tumor volume) primary tumor delineations (GTVp), manually annotated by radiation oncologists, were provided as bounding boxes extracted around the oropharynx, with a dimension of 144x144x144 pixels. Sequences of three consecutive slices were extracted from each volume, as described in [3], as the method is trained and validated on sequences.

For the current project only PET and GTV were used. As the model in [3] is trained on sequences, PET sequences were extracted. One sequence consists of three consecutive PET slices. Figure 4 shows an example of created sequences.

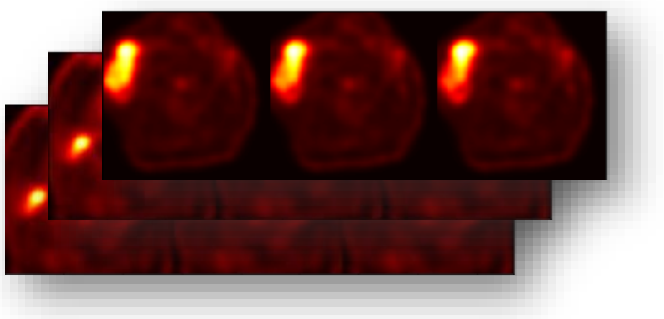

Figure.4. Example of PET sequences used in the study.

2.2 Cluster analysis

Cluster analysis is a statistical method used to processed data. It organizes objects into groups or clusters based on their close association.

Cluster analysis is concerned with data matrices in which the variables have not been partitioned beforehand into criterion versus predictor subsets. The objective of cluster analysis is to find similar groups of subjects, where “similarity” between each pair of subjects means some global measure over the whole set of characteristics [24].

Cluster analysis is an unsupervised learning algorithm. Contrary to many other statistical techniques, cluster analysis is frequently employed when no assumptions are made regarding the probable relationships among the data. Although it tells us where relationships and patterns in the data are, it doesn't explain what they might be or what do they mean. Unsupervised learning methods work well for data imbalance problems. There are several

steps needed to perform cluster analysis (Figure 5). Each step is described in detail below.

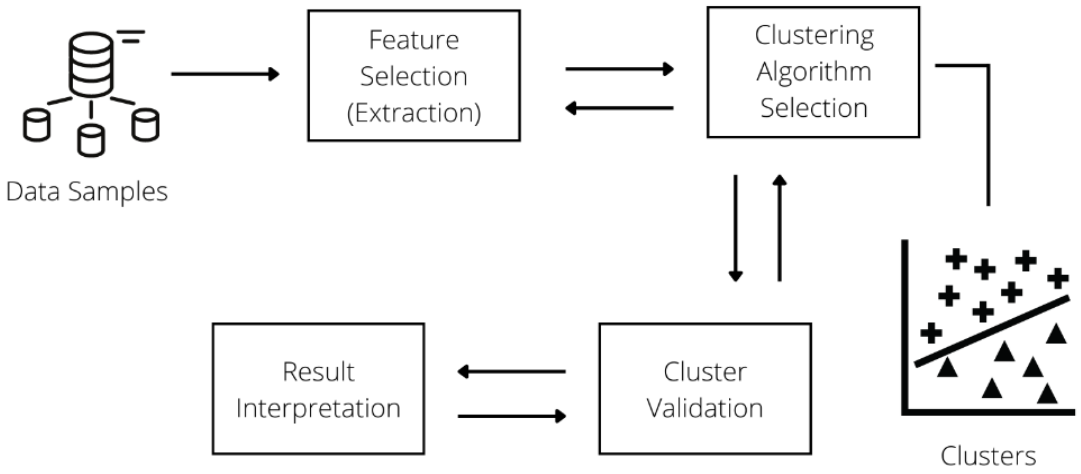

Figure.5. Steps for cluster analysis. Image adapted from [25].

2.2.1 Features selection (extraction)

To perform clustering analysis, features which could be representative of each data sample were investigated. Maximum, minimum, mean, standard deviation and mean entropy were calculated using pixel SUV values of PET sequences. In addition, a histogram analysis was performed and parameters such as kurtosis and skewness were also included.

In the formulas below we will assum that a sequence is a matrix *F* with dimension 432x144 which has been flattened to a one dimension array f. The mean standardized uptake value (meanSUV) is the average value of FDG uptake activity in an area. The maximum standardized uptake value (maxSUV) represents the value of the pixel with the highest FDG uptake activity. The minimum standardized uptake value (minSUV), represents the value of the pixel pixel with the lowest FDG uptake activity[26]. The standard deviation (std) [27] measures how data is dispersed relative to its mean. It is calculated as the square root of its variance (1).

$$std = \sqrt{\frac{\sum_1^N (f_i^2 - meanSUV^2)}{N}} \quad (1)$$

N – number of pixels in the PET sequence, f_i – an i-element of the array f.

The entropy was also considered as a relevant feature. It is a measure of randomness. Inhomogeneous textures have low entropy, whilst homogeneous textures have high entropy [28]. Entropy is calculated with the help of creation of the gray level co-occurence matrix(2) (see Figure 6):

$$entropy = -\sum_{i=0}^{n-1} \sum_{j=0}^{n-1} p(i,j) \quad (2) \text{ [29]}$$

n – number of gray levels, $p(i,j)$ – probability of two pixels separated by specified offset having intensities i and j.

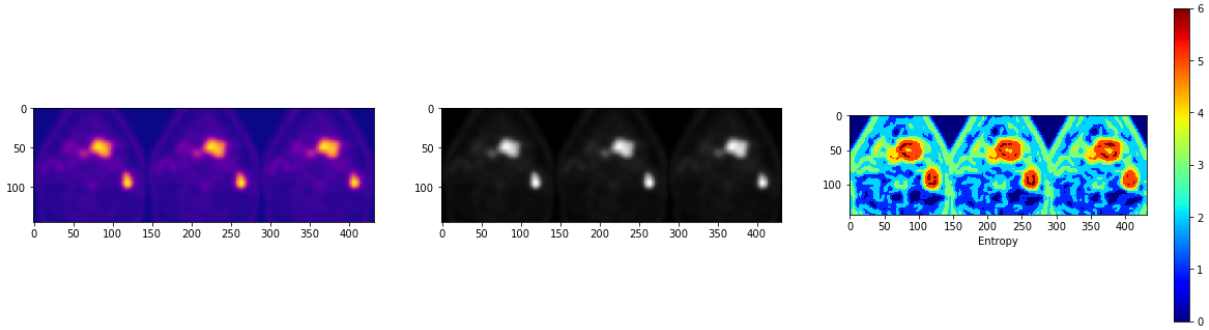

Figure 6. Representation of calculated entropy.

Histogram analysis is also a way of further analyzing PET images and obtaining imaging biomarkers [30]. Therefore, skewness and kurtosis were calculated.

Skewness describes how much the data distribution is asymmetrical from the normal distribution [28]. Skewness is calculated as adjusted Fisher-Pearson coefficient of skewness(3)[31]:

$$skewness = \frac{\sqrt{N(N-1)}}{N-2} \frac{\sum_{i=1}^{N} (f_i - meanSUV)^3 / N}{std^3} \quad (3)$$

N – number of pixels in the PET sequence, f_i – an i-element of the array f

When the most values are concentrated on the left of the mean the skewness is positive. When the most values are concentrated on the right of the mean the skewness is negative (see Figure 7). If skewness is equal to 0, it is a symmetrical distribution around the mean [28].

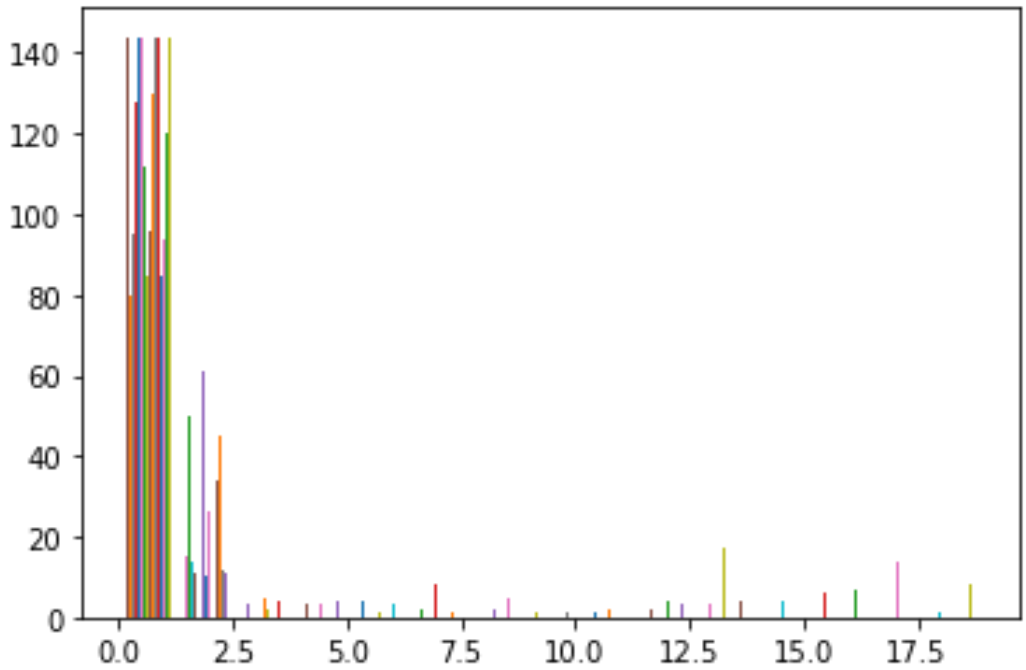

Figure.7. Representation of negative skewness for one of the PET sequences.

Kurtosis is a statistical measure that describes how much a distribution's tails differ from the tails of a normal distribution. It shows 'peakedness' of a data distribution(4) [28].

$$kurtosis = \frac{\sum_{i=1}^{N}(f_i - meanSUV)^4/N}{std^4} \quad (4)\ [31]$$

N – number of pixels in the PET sequence, f_i – an i-element of the array f

When the value for kurtosis is below 3, the data distribution is sharper than a normal distribution, with values concentrated around the mean and thicker tails (See Figure 8). When it has values above 3, the data distribution is flatter than a normal distribution with a wider peak, the probability for extreme values is less than for a normal distribution and the values are spread more widely around the mean. Kurtosis with a value equal to 3 shows a normal distribution of the data [28].

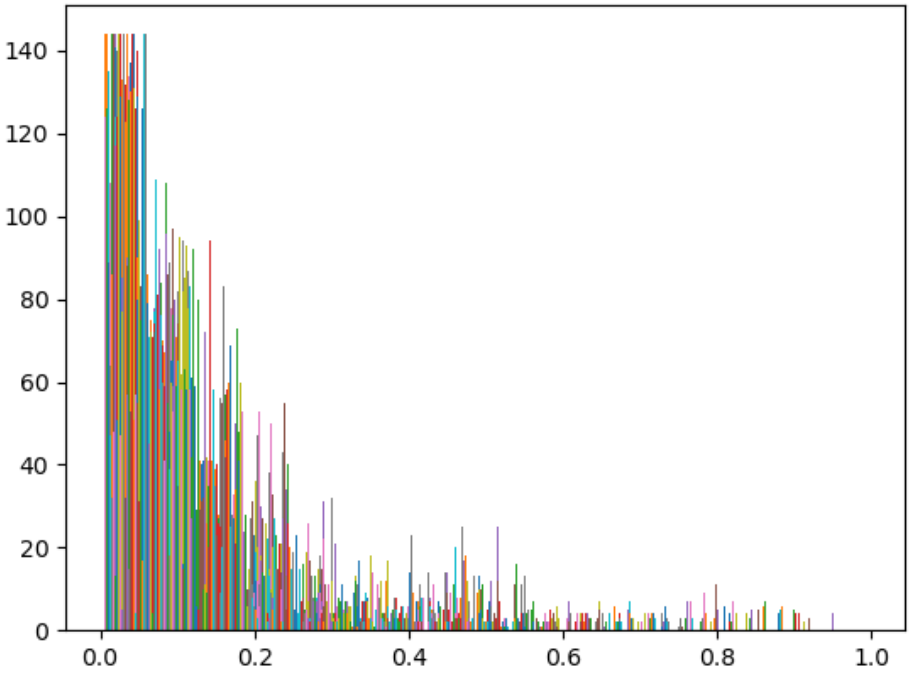

Figure.8. Representation of kurtosis with the high probability of extreme values from an example PET sequence.

All the features described above were calculated in the programming language Python using the Peregrine high performance computing cluster of Center for Information Technology of the University of Groningen. The parameters like mean SUV, max SUV, min SUV, std and entropy were calculated with the help of the python library NumPy. Histogram analysis and its features as skewness and kurtosis were performed with the help of SciPy python library. For visualization of some features the python library

Matplotlib was used. After all the features were selected and calculated for all the sequences, a dataframe was created using Pandas python library.

2.2.2 Clustering algorithm selection

Two different clustering algorithms were selected for this study: K-Means and Fuzzy C-means. Both methods are described in the next sections.

2.2.3 K Means clustering

The K-Means algorithm assigns data to the clusters by trying to separate data samples in n groups of equal variance and minimizing a criterion known as the inertia or Within-Cluster Sum-of-Squares (WCSS) [36]. This algorithm requires defining the number of clusters manually. It scales well to a large number of data samples and it has been used across a large range of applications in many different fields. It is very common among the applications in the medical imaging field. Figure 9 shows how the K-means algorithm works.

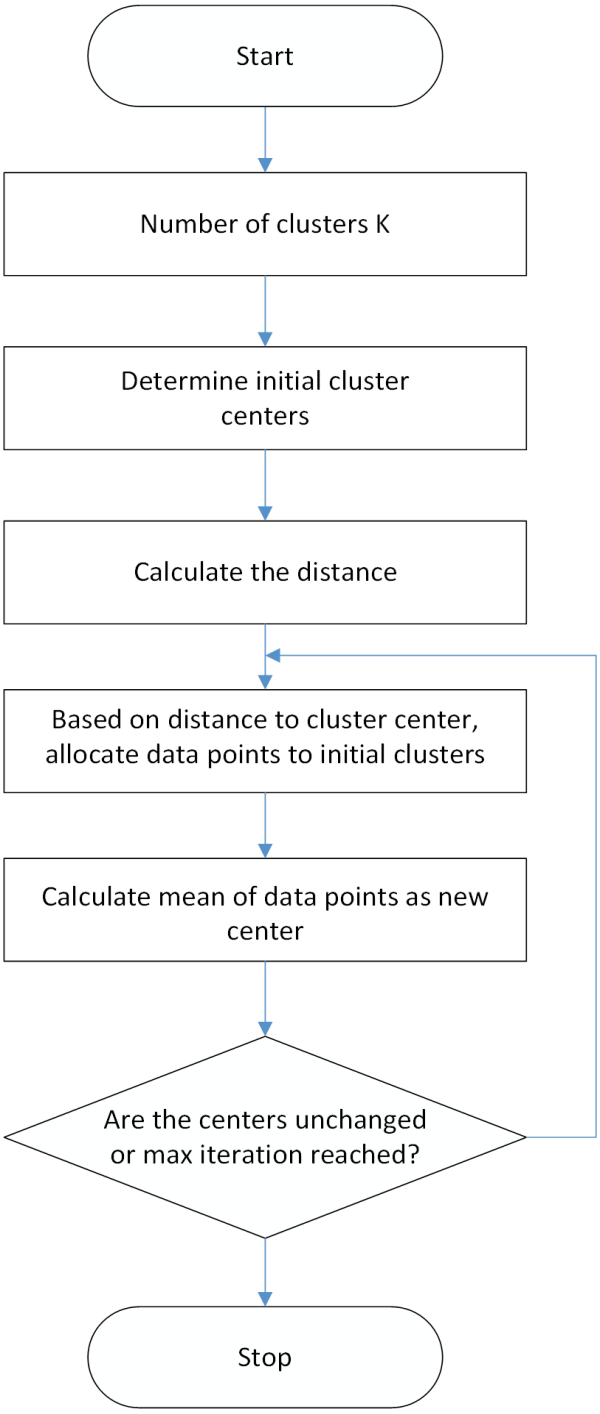

Figure.9. K-means clustering algorithm. Image adopted from [32].

The distance used in this study was the ‘Euclidean Distance’ (5) and it is calculated between the cluster center and each data point within the cluster. Consequently the mean distance of all the data points within a cluster is calculated and centroids are formed [33]. The idea behind the K-means

clustering algorithm is to minimize the distance between cluster center and the data points.

$$d(f,c) = \sqrt{\sum_{i=1}^{n}(f_i - c_i)^2} \quad (5)$$

n – number of features extracted from the PET sequence, f and c – coordinates of the data point and the cluster center in the multidimensional space, fi and ci – Euclidean vectors, starting from the origin of the multidimensional set of features (initial point).

The number of clusters was chosen according to the elbow method. It uses the WCSS criterion for selecting the optimal number of clusters k [34]. It was calculated with the formula below (6):

$$WCSS = \sum_{k=0}^{k}\sum_{n=0}^{d_n}(distance(d_i, {C_k}^2) \quad (6)$$

C – cluster centroids, *d* – data points in a cluster, *k* – number of clusters, *n* – number of features extracted from the PET sequence.

The ideal number of clusters k was chosen according to the plot showing the relation between WCSS and the number of clusters. The optimal k was selected where WCSS curve started to band and formed an ‘elbow‘ [35].

K-Means clustering is good to use in the medical imaging field because it is simple to implement, it is scalable and performs quickly with the huge datasets of medical images and has a good generalization of clusters for different shapes and sizes. However, this algorithm is sensitive to outliers and with the increasing number of dimensions its scalability decreases. Also, it is time consuming to choose the number of clusters k manually [35].

2.2.4 Fuzzy C mean clustering

Fuzzy C-means (FCM) is a method of clustering which allows one datasample to belong to two or more clusters: each data point has a degree of membership (probability) of belonging to each cluster [37]. Figure 10 shows how the Fuzzy C-means algorithm works. This algorithm is based on the minimization of the objective function (7):

$$J = \sum_{i=1}^{K} \sum_{j=1}^{N} \mu_{ij}^{m} \left\| f_j - c_i \right\|^2 \quad (7)$$

ci – cluster center, N – number of features extracted from the PET sequences, fj – position of the datapoint, K – number of clusters, m – fuzzification coefficient of the algorithm, μ - representative matrix for the membership of each element in each cluster

The number of clusters was chosen according to the Elbow methos as for the K-means clustering.

The cluster center matrix V_k is calculated as (8):

$$V_k = \frac{\sum_{i=1}^{N} \mu(i,k)^m f_i}{\sum_{i=1}^{N} \mu(i,k)^m} \quad (8) \text{ [38]}$$

μ – representative matrix for the membership of each element in each cluster, m – fuzzification coefficient of the algorithm, N – number of features extracted from the PET sequences.

The representative matrix for the membership of each element in each cluster μ is calculated with the formula (9):

$$\mu(i,k) = \left(\sum_{j=1}^{C} \left(\frac{D(i,k)}{D(i,j)}\right)^{\frac{2}{m-1}}\right)^{-1} \quad (9) \text{ [38]}$$

C – number of clusters, m – fuzzification coefficient of the algorithm, D – squared distance between the data point f_i and the cluster center V_k.

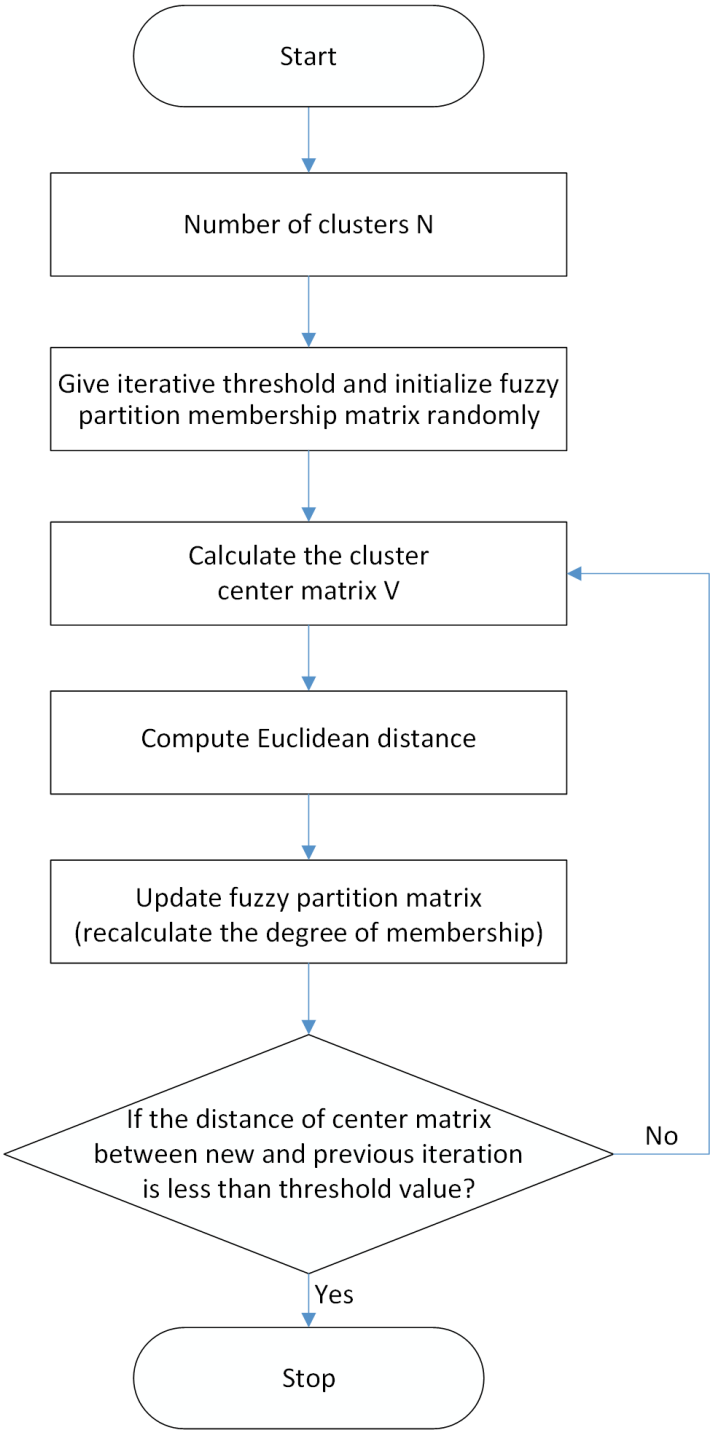

Figure.10. Fuzzy C-means clustering algorithm. Image adopted from [36].

Fuzzy C-means clustering algorithm gives best results for overlapped data set and comparatively better then k-means algorithm. Unlike k-means where the data point must exclusively belong to one cluster centroid where the data point is classified, Fuzzy C-means allow the data point belongs to

several clusters. Another advantage is membership to each cluster center as a result of which data point may belong to more than one cluster center. However this algorithm is more time consuming than k-means clustering [37].

2.2.5 Cluster validation

After the implementation of both clustering algorithms they were validated according to the parameters that are useful for current methods. For K-Means clustering the representative measure is distance between cluster center and the closest data point. There were nine different model fitting provided with nine different combinations of features of PET sequences. Also there was an additional experiment conducted with the data normalization. Data was recorded into a comparison table. For the Fuzzy C-means there was only one experiment conducted. The result was compared to the performance of K-Means clustering algorithm. Finally, the observation of findings and conclusions were made.

2.2.6 Result interpretation

In the current project there were two methods of the interpreting the results of clustering used: qualitative and quantitative.

To perform a qualitative analysis, a scatter plot was created with the cluster division representation. Some of the sequences belonging to each cluster were picked and displayed, conclusions were made. The observations considered were as follows: whether the clusters were relevant, if there were any similarities between sequences belonging to a same cluster, if there were any significant differences between sequences belonging to different clusters.

For quantitative analysis we used metrics typically used for the different clustering algorithms. K-means clustering algorithm was applied to different sets of calculated features. To determine the best combination the distance between cluster center and the data point within the cluster was extracted and compared. The comparison table of all the distances was created. Finally, the performance of the K-means clustering algorithm and the Fuzzy C-means algorithm was discussed.

2.2.7 Sequence selection and method evaluation

To evaluate the quality of the different clustering methods, sequences were selected to train and validate the 2D model from [3]. All the PET features were calculated for the new training (124 patients) and validation (32 patients) sequences and were recorded into two dataframes. The best performing K-means clustering and Fuzzy C-means trained models were used to predict the clusters of the new data.

For training the models all sequences containing tumor pixels for 2.5% or above of their area were included as positive sequences.. The new training and validation datasets had to include 80% of sequences with tumor and 20% without tumor. The 20% of negative sequences were selected considering an equal number of sequences from each assigned cluster. Finally, 80% of positive sequences and 20% of negative sequences were merged into one dataframe for further model training. Results were evaluated by metrics: train and validate loss functions, dice score coefficient, precision and recall.

3 Results

In this section, results from some of the most promising clustering experiments were reported. In the first part the features extraction and dataset description is provided, also clustering algorithms performance is described. In the second part a report on slice selection is given. In the last part of this chapter the results from the experements on the automatic segmentation model were provided.

3.1 Dataset description

After all the features were selected and calculated, a dataframe was created using Pandas library (Table 1). It took around 63 hours to populate the dataframe with the extracted features from all the sequences. Each row corresponds to one of the 12922 extracted sequences. In the columns the ID of the corresponding patient, the number of the slice from which that sequence starts, and the seven calculated features were reported.

Table 1. Dataframe containing the extracted features

	ID	slice	meanSUV	maxSUV	minSUV	std	skewness	kurtosis	entropy
0	Pat99	138	2.554469	8.801387	0.002295	2.433752	0.522430	-0.977176	1.713461
1	Pat88	90	1.061445	8.077387	0.002461	0.815599	0.644698	0.417928	2.569637
2	Pat45	28	0.744742	4.204774	-0.000119	0.594423	0.190037	-0.422159	2.364535
3	Pat47	108	1.376534	7.264812	0.000430	0.970712	0.900959	0.313343	1.769467
4	Pat28	52	0.549582	3.220131	0.000094	0.566837	0.403938	-0.860266	2.432163
...	...	...	...	...	...	...	...	...	...
12917	Pat54	121	1.874440	10.683081	0.000822	2.244467	1.257937	1.354259	2.179173
12918	Pat88	64	0.678714	2.391438	0.000360	0.530995	0.108706	-0.685870	2.628472
12919	Pat64	111	0.961277	5.035169	0.000334	0.744614	0.814702	0.129890	2.448302
12920	Pat40	66	0.683256	4.851300	0.000361	0.534812	0.546865	0.632192	3.311005
12921	Pat55	118	1.394106	7.497525	-0.000116	1.128693	0.420491	0.293038	2.216757

12922 rows × 9 columns

To see if a correlation between the calculated features existed, a sub-scatter plot was created (see Figure 11). There is a color division between sequences containing tumor and not containing tumor: blue – no tumor sequences, orange – tumor sequences.

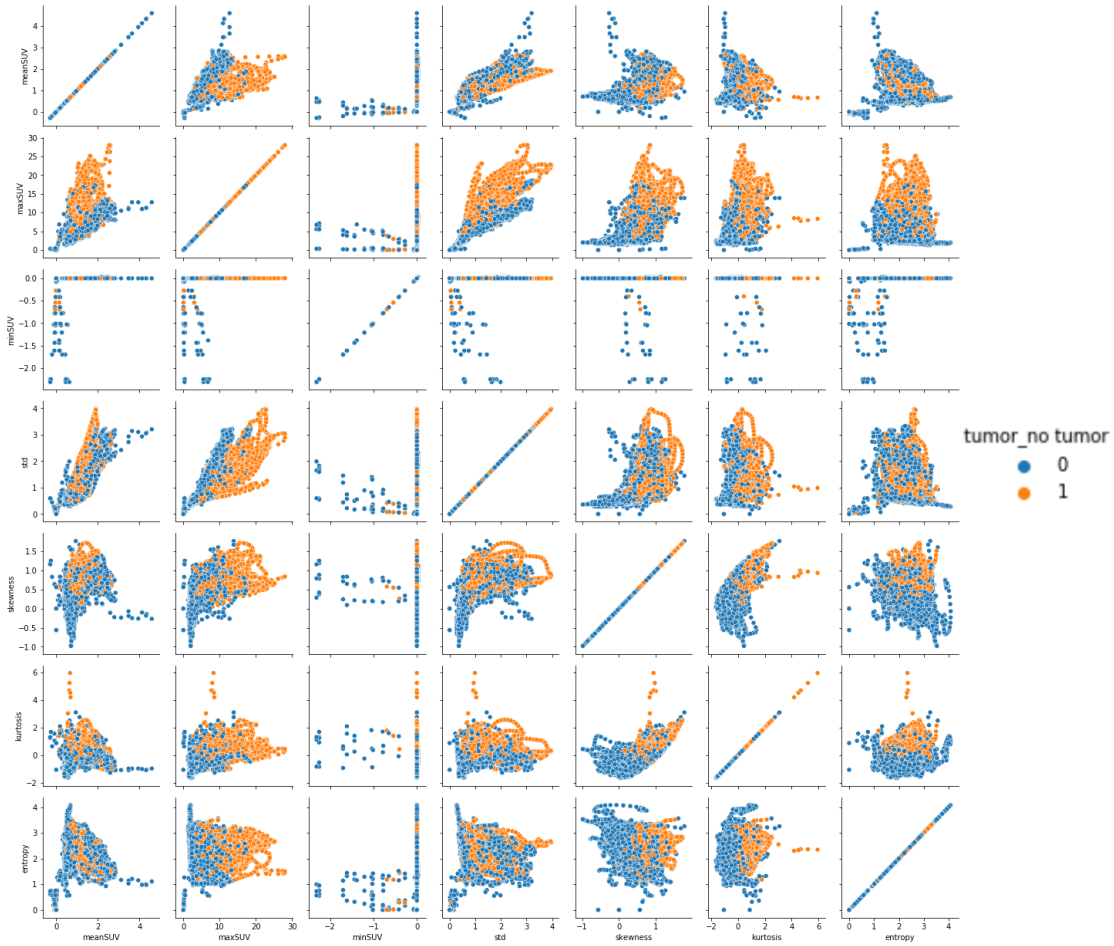

Figure.11. Sub-scatterplot of all extracted features from the PET sequences.

3.2 K-Means clustering

Nine experiments were conducted with nine different combinations of features extracted from PET sequences. In addition, one experiment was conducted using data normalization. First of all, to create clusters with the K-means clustering algorithm it the optimal number of clusters needed to be defined. The Elbow method was used nine times for each experiment. The

optimal number of clusters for all the models in the current project was 3, it is the point where the curve bands and starts to flatten. Figure 12 shows an example of the relation between WCSS and the number of clusters for the K-means clustering model trained on all the extracted features from the PET sequences.

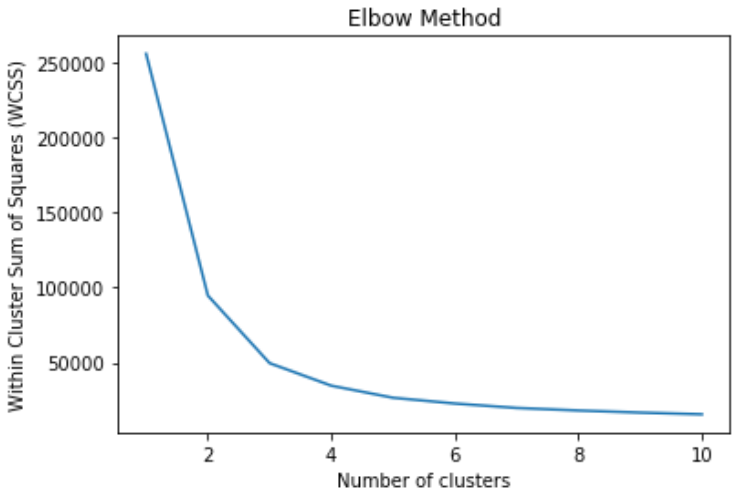

Figure 12. An example of the relation between WCSS and the number of clusters.

After the optimal k was defined, the clusters were created. Figure 13 shows the distribution of the sequences among the created clusters when using all the extracted features. It is visible that the clusters are not equal in the number of data points.

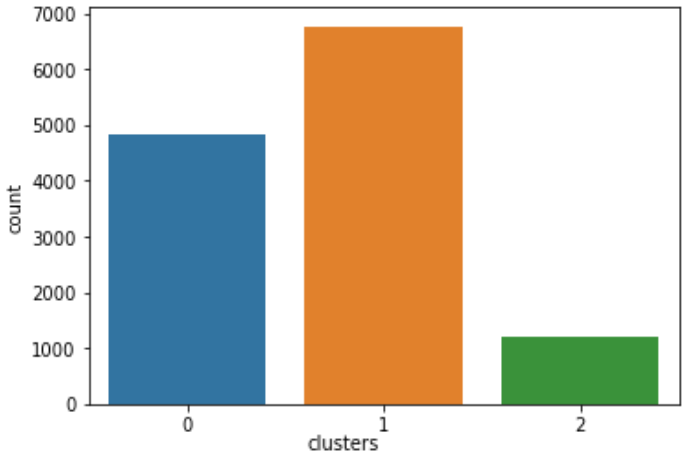

Figure 13. The bar chart of the data distribution among the created clusters using the K-means algorithm and all the extracted features.

In Table 2 a quantitative method to evaluate the performance of the K-means clustering algorithm is reported. It contains nine different most relevant combinations of the extracted features for training the K-means clustering model. The evaluation of the performance of the clustering algorithm consists of the comparison of the distances between cluster centroids and sequences within the clusters on raw training data, on the normalized training data and on the training and validation data used for the deep learning segmentation model.

Table 2. Comparison table of cluster distances, normalized clusters distances, segmentation training cluster distances and segmentation validation cluster distances

Features used	Number of clusters, k	Distance	Normalized clusters, distance	Segmentation training, distance	Segmentation validation, distance
All	3	49616.71	55847.30	82814.74	20224.44
MeanSUV, maxSUV	3	36799.92	7413.95	10394.61	2792.89
No std	3	47216.69	49401.25	72436.11	17903.22
No skewness	3	48498.25	49009.67	73865.10	17424.85
Entropy, maxSUV	3	38030.25	9932.85	53803.32	3595.24
MeanSUV, kurtosis	3	3019.80	8189.53	7851.42	2970.99
No minSUV	3	49538.50	43054.67	75888.53	16063.85
No std, meanSUV	3	45279.09	42008.90	69513.03	15463.65
Skewness, kurtosis	3	2122.57	7912.12	5861.34	2864.53

Normalization is an important pre-processing step of clustering. It helps to scale large and small values in the dataset so that each variable can have the same range. It can help to improve the efficiency of the clustering algorithm [39]. For this reason, the extracted features were first normalized and the clustering model was trained. The comparison table shows that for the combination of features as for example meanSUV and maxSUV after

normalization the distance was shorten. It is because the range of values between the two variables was quite different before normalization.

Other K-means clustering models using normalized features were trained and the ones with the shortest distances used: all features, all features except std, meanSuv and maxSUV, skewness and kurtosis, entropy and maxSUV, and meanSUV and kurtosis.

Afterall, the performance of the created K-means clustering models was tested on new patients used to train and validate the deep learning segmentation model. In Table 2 we can observe that the distance in training dataset became larger compare to the clustering on the initial dataset and in the validation dataset – smaller.

It is difficult to visualize the clusters using all seven features at the same time in one multi-dimensional scatter plot. For this reason, few scatter plots were created using two different variables per time to visualize the created clusters. In this section there were two visualized results presented. Figure 13 shows the scatter plot of the created clusters using the entire set of extracted features visualized using meanSUV versus maxSUV. It is possible to see a clear division between the created clusters on the y axis.

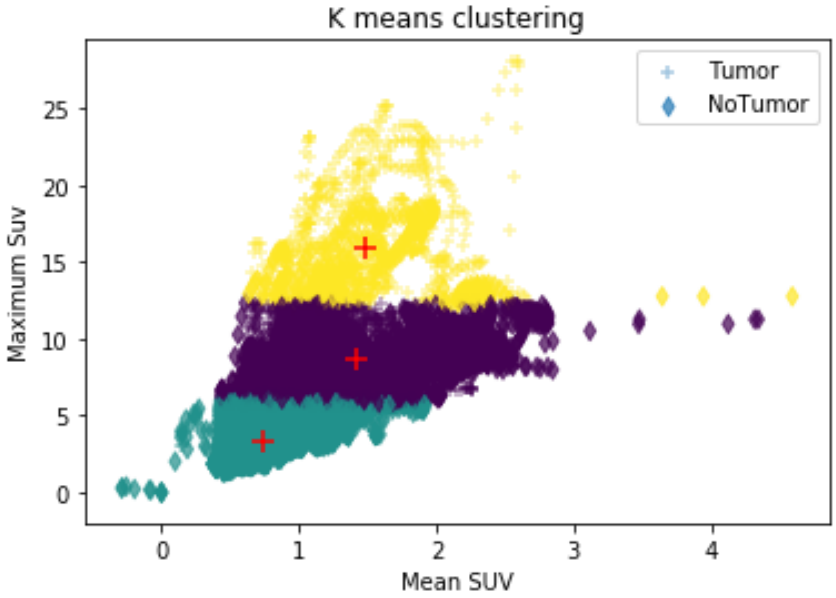

Figure 14. Scatter plot of meanSUV versus maxSUV showing the clusters created using the K-means clustering algorithm trained using all the extracted features.

To perform qualitative analysis few sequences were plotted for each created cluster. Figure 15 shows sequences belonging to each cluster (from 1 to 3). The clusters from Figure 14 that are from bottom to top are in the order from left to right in Figure 15.

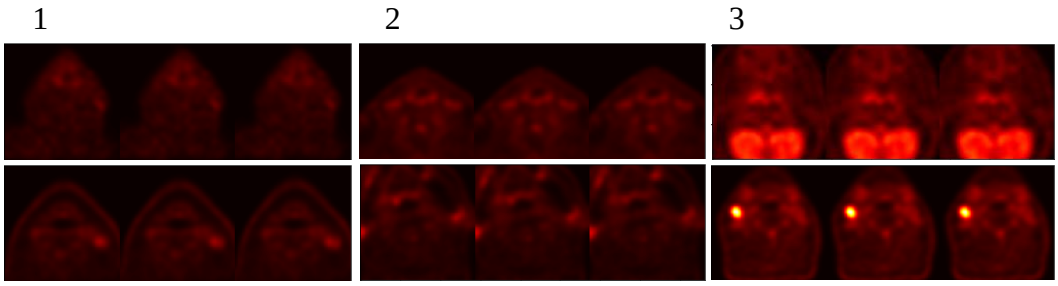

Figure 15. Samples of sequences belonging to the created clusters using K-means clustering algorithm trained using all the extracted features. . Selection based on the meanSUV - maxSUV visualization.

The visualization of the sequences showed to be consistent with the clusters. It is possible to distinguish the difference between pixel intensities in each cluster visually. Higher maxSUV values correspond to brighter areas on sequences.

The visualization that was not showing a clear distinction between clusters was also reported. Figure 16 is a scatter plot of kurtosis versus entropy showing the same created clusters as before. The created clusters are overlapped in the figure. This is an issue of visualization as the clusters were created on more than these two features.

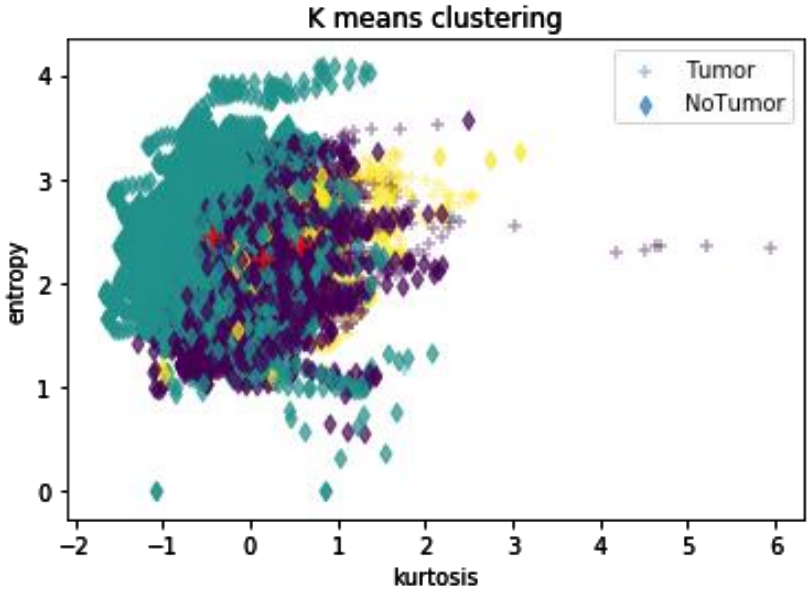

Figure 16. Scatter plot of kurtosis versus entropy showing the clusters created using K-means clustering algorithm trained using all the extracted features.

For the qualitative analysis few sequences were plotted for each created cluster to explore the results of the K-means clustering model visualized with two different features. Figure 17 shows sequences belonging to each cluster (from 1 to 3). The clusters from Figure 16 that are from left to right are in the same order in Figure 17.

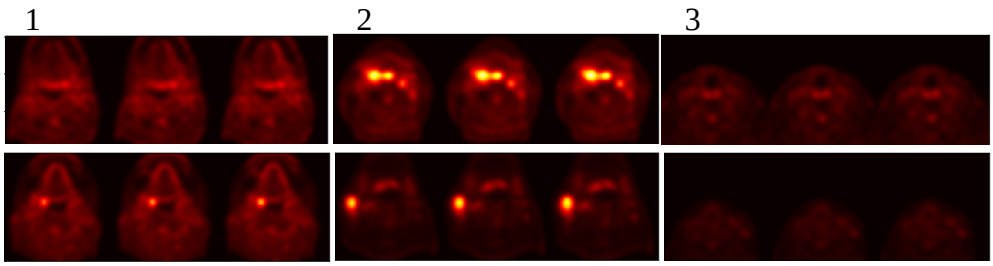

Figure 17. Samples of sequences belonging to the created clusters using K-means clustering algorithm trained using all the extracted features. Selection based on the kurtosis-entropy visualization.

3.3 Fuzzy C-means clustering

The Fuzzy C-means algorithm using normalized features was aslo implemented. The number of clusters was defined using the same Elbow method as described in the previous section. The optimal nuber of clusters was 3, as well as for the K-means. Figure 18 shows the distribution of the sequences among the created clusters. The data was not equally distributed among clusters as well, however differently from the K-means.

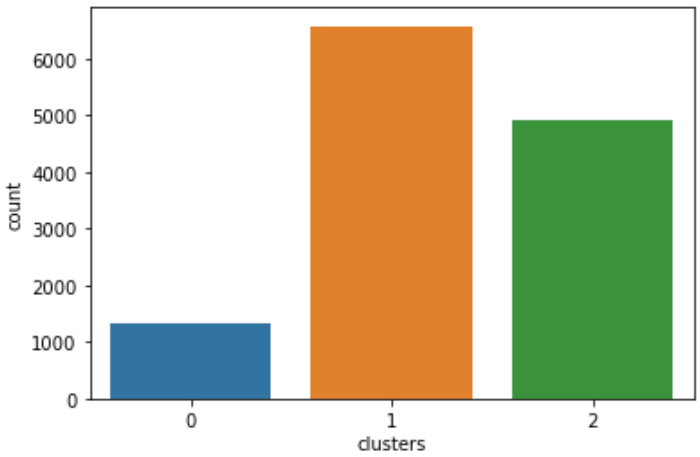

Figure 18. The bar chart of the data distribution among the created clusters with the Fuzzy C-means algorithm and all the extracted features.

Figure 19 shows the scatter plot of the created clusters using the entire set of extracted features visualized using meanSUV versus maxSUV.

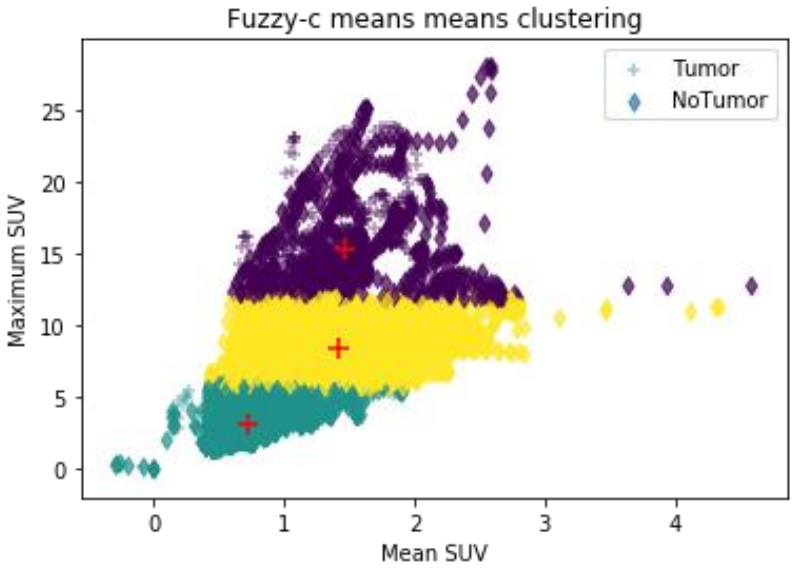

Figure 19. Scatter plot of meanSUV versus maxSUV showing the clusters created using the Fuzzy C-means clustering algorithm trained using all the extracted features.

The division between clusters in the scatterplot of meanSUV versus maxSUV for Fuzzy C-means and K-means is similar, however the cluster centers are slightly different (see Table 3).

Table 3. Comparison of cluster centers in K-means and Fuzzy C-means algorithms

	K-Means		Fuzzy-c means	
Clusters	meanSUV	maxSUV	meanSUV	maxSUV
1	0.74215442	3.28381926	0.71780598	3.09893199
2	1.41720731	8.64085724	1.41402938	8.40373263
3	1.47677749	15.91599427	1.47078422	15.39765794

For Fuzzy C-means clustering it is possible to assess the quality of the cluster division using the same sequences as in Figure 15.

3.4 Sequence selection

A new set of patients was used to test the clustering models from the previous sections. The dataset contained 155 patients (124 in the training dataset and 31 in the validation dataset). All the features were extracted from the new PET sequences and recorded into two dataframes. The training dataset consisted of 17565 sequences containing and not containing tumors. The validation dataset had 4544 sequences. Sequences where tumor pixels were more than 2.5% of the entire sequence area were 1561 in the training dataset and in 702 in the validation dataset. Hence, the clustering methods for proper sequence selection from the negative class were tested on 16004 sequences in the training dataset and on 3842 sequences in the validation one. To select the 20% of the negative class, an equal amount of sequences were selected from each cluster.

Finally, 7 dataframes were created for the training s and 7 dataframes for the validation dataset (6 dataframes as a result of K-means clustering and 1 as a result of Fuzzy C-means clustering). A total of 1951 sequences were used for training and 878 for validating. These numbers were created combining sequences containing tumors and sequences selected with the proposed methods. The dataframes included patient IDs and number of slices from which that sequence begins.

3.5 Method evaluation

After the new training and validation dataset were created as explained in the previous section, the model from [3] was trained and evaluated to assess the performance of the method proposed in the study. All models were trained on sequences extracted from 124 patients and validated on 31 patients. Model 1 was used as benchmark. It was trained on sequences selected as explain in [3]. The data split was created selecting all sequences containing a percentage of tumor above 2.5% (representing the 80% of the

entire dataset), a small percentage of sequences with tumor below 2.5% (15% of the entire dataset), a small percentage of sequences containing no tumor (5%).

Training and validation loss functions for all models trained are reported in Figure 20 and Figure 21. From these images it is possible to say that the created models are more instable during training, most of the models have spikes during the entire training process. The validation loss function decreases for all newly created models. Most of the models have spikes in the validation loss function as well. In some models spikes become smaller with the time.

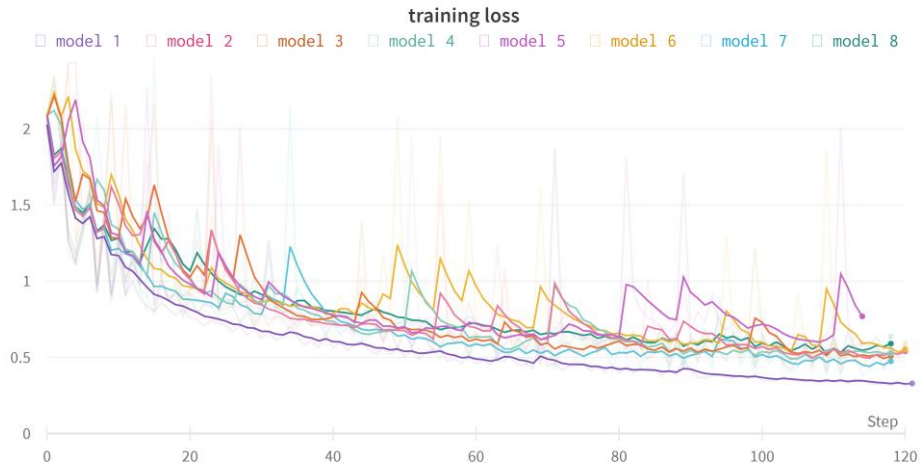

Figure 20. Training loss function for all trained models.

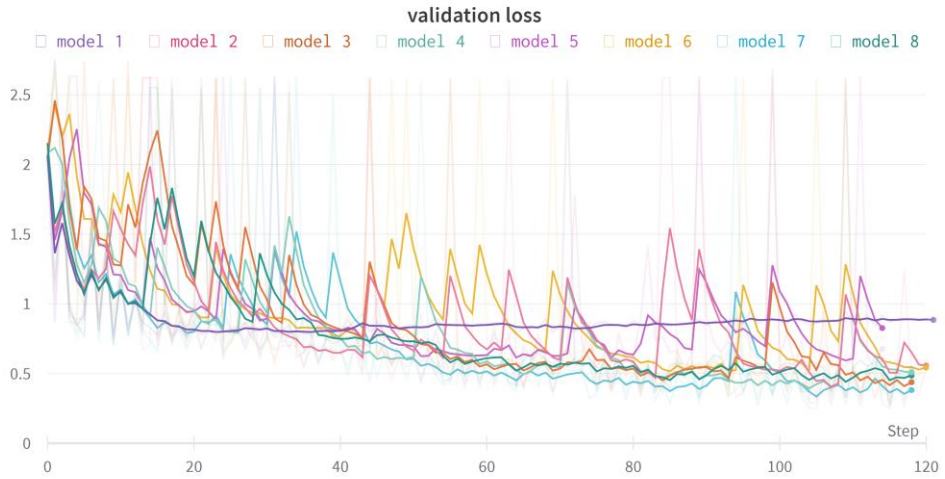

Figure 21. Validation loss function for all trained models.

During training on sequences precision, recall and mean dice coefficient are calculated on the validation set made of selected sequences (see Figures 22-24). The different models were trained for a maximum of 150 epochs. The selected checkpoint was the one corresponding to the lowest value of validation loss after 100 epochs. Lastly, the selected checkpoint of each model was used to create predictions on all sequences contained in the 144x144x144 pixels volumes [3]. Reconstructed predicted volumes were evaluated using precision, recall and mean dice score coefficient (DSC) at different probability thresholds (see Appendix 1).

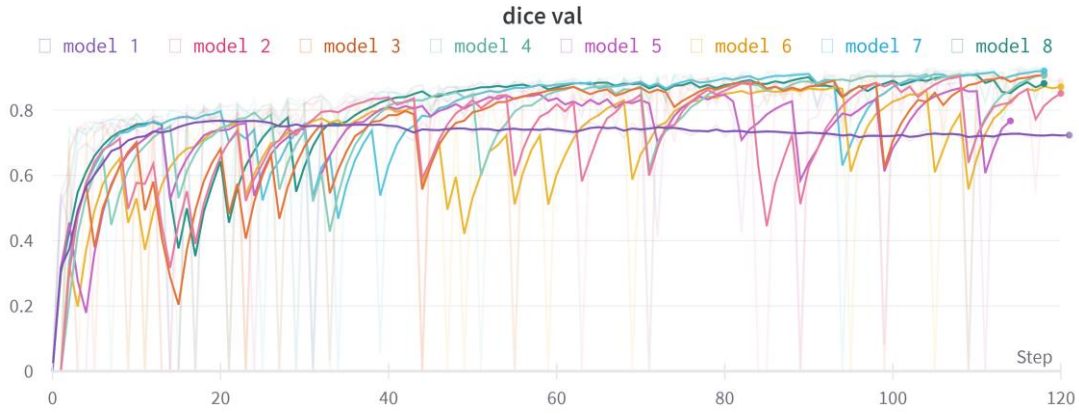

Figure 22. Validation mean dice score coefficient during training.

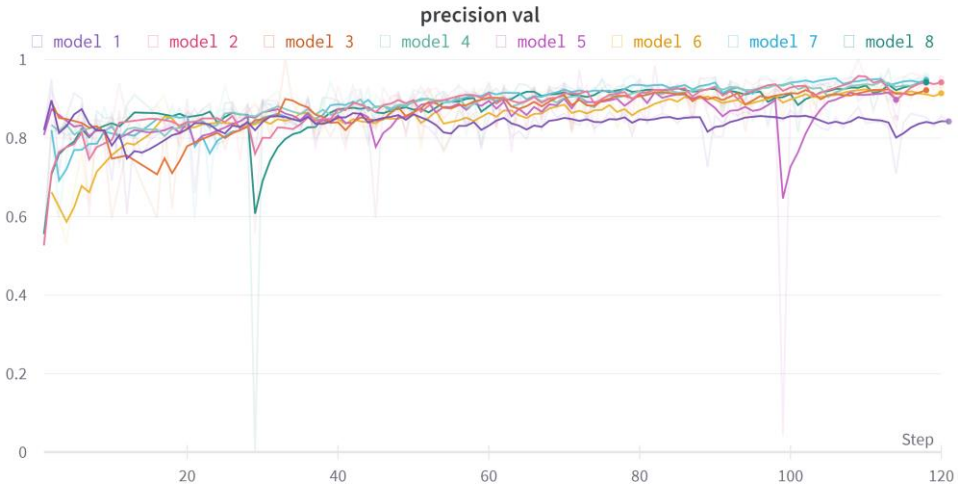

Figure 23. Validatiom precision during training.

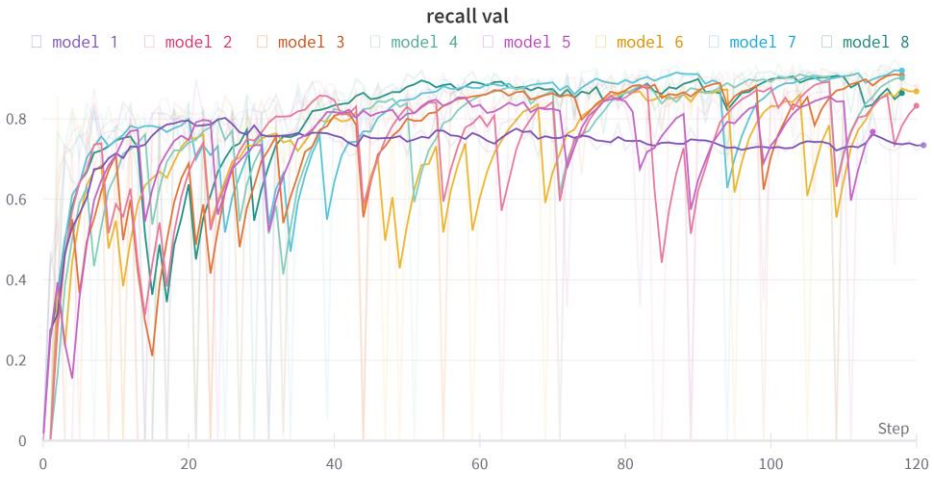

Figure 24. Validation recall during training.

According to the recorded metrics, the best performing models compared to the reference model (Model 1) were Model 2 (K-means clustering was trained on all extracted features) with the mean DSC: 0.619, precision: 0.629 and recall: 0.959 and Model 8 (Fuzzy C-means was trained on all extracted features) with the mean DSC:. 0.616, precision: 0.608 and recall: 0.949. The mean DSC of the Model 2 improved by 17.5%, precision was improved by 37.1% and the recall by 1.3%. In the Model 8 metrics were improved by 16..9%, 32.5% and 0.2% respectively.

4 Discussion

In this project unlike in the work [3], the sequences for the 2D segmentation model were selected with the help of the undersampling clustering technique. Therefore, at the beginning of this chapter the methods used in this project were discussed and in the second section the results of the experiments and their evaluation. Also, an ethical considerations section was included at the end of this chapter.

4.1 Methods

Class imbalance is a quite challenging and widespread problem. Hence, exploring methods for its solution is an essential task. The choice of using undersampling clustering technique was made because it is easy to implement, quick in performance and showed promising results in the medical imaging field [16,17].

The features that were extracted from the PET sequences showed different results in terms of division between tumor and not tumor sequences. It was visible that when using maxSUV as feature to visualize datapoints the division between tumor and not tumor sequences was quite clear and it was possible to assume that clusters could be created using this variable. Some of the extracted features showed overlapping tumor and not tumor sequences. After the clustering methods were implemented, we saw that some of the features were not useful to use for cluster formation. The scatter plot in Figure 11 also visually confirmed it. Moreover it was time comsuming to calculate all seven features for all the PET sequences, which can be considered as a limitations of the chosen methods.

The outcome of the implemented K-means and Fuzzy C-means clustering was interesting. It is difficult to visualize the clusters using all seven features at the same time in one multi-dimensional scatter plot.

Therefore, the scatter plots were created using two features per time. There were few different combinations of features selected to vizualize the created clusters. In cases where overlapping clusters were observed, one possible alternative way to observe the clusters could have been using at least a three-dimentional space. Comparing the sequences picked from every cluster (Figure 15) it was possible to see the difference between clusters as well as similarities in pixel intensities inside of the cluster. Since the clusters were visualized based on maxSUV, higher values showed pixels with higher intensity on the sequences. Comparing the distances between cluster center and the datapoint within the cluster some useless features were identified. We discarded cases where the distance in the experimements containing these features for clusters creation was the highest and the qualitative analysis was barely possible (for std and minSUV, for instance). The distances calculated for data points of the training and validation dataset used for the segmentation model became larger in the first case and lower in the second case compared to the clustering training data. One possible reason could be the difference in the amount of patients. More patients included to train the segmentation model means that the variability in this dataset is much higher. For the validation dataset the variability is lower because there are less patient in this dataset compare to the initial dataset.

The Fuzzy C-means clustering method was not as detailed explored as the K-means one, because of the limited time on the current project. However, some interesting conclusions were still made. The visualization of the created clusters using the same features looked similar to the K-means. The distribution of the data among the created clusters was different, but not equall es well. When the clustering models were fitted to the newly created training set for sequence selection, the distribution of the data among clusters was quite equal.

4.2 Results

To evaluate the performance of the chosen method, the model from [3] was trained on the datasets created using cluster analysis. The results were promising. Looking at the training and validation loss functions (Figure 20-21) is is possible to say that the network was learning (descending curves), even though during training the models are more instable than the reference one. In the valitation loss function the reference model stabilized after 20 epochs, however, for the models that used the new splits of the data the function is decreasing and for some models the function stabilized (the existing spikes either become lower or disappear). Perhaps, if the models were trained with more epochs, the performance would be even better and more stable. Regarding the DSC, precision, and recall graphs (Figure 22-24), the newly created models have higher values compared to the baseline model. We can see that the precision is more stable than the recall for all the models. Finally, the most promising models were Model 2 and Model 8. All used metrics were improved compared to baseline Model 1. Also, the curve for the plotted validation mean DSC, precision and recall during training had higher values on the y axis. Model 8 looked more stable than Model 2 as it contained fewer unwanted spikes. It could mean that Fuzzy C-means method performed well.

4.3 Ethical considerations

It is important to do research considering all possible ethical aspects. In the Netherlands the number of patients having head and neck cancer increased by 55% in the last years [40]. Moreover, the head and neck region contains a lot of important organs and complex structures that are important not to be dagamaged while cancer treatment. Hence, it is important to segment the tumor area from PET/CT scans for treatment, so the 2D automatic tumor segmentation model [3] needs to be improved. To do so,

sequences for training have to be selected correctly. A more accurate automatic segmentation model could be a good support for radiologists in their workflow.

The anonymity of patient data is well maintained in current project. All patients used in this study gave informed consent before their data was used. PET/CT scans involved in the research consist of protected and anonymized data of patients in special identification codes [41]. The implemented code was made with the consideration of data security and protection.

It is important from an ethical perspective to make sure that the results are correct and real. All used sources of information are either cited or mentioned in the project to avoid plagiarism.

5 Conclusions and future work

In the current project the undersampling clustering-based method for sequence selection was explored. It showed promising results and has a potential for further work.

Seven features describing PET imaging were selected in order to create clusters of sequences. The dataframe containing the features was created. Two clustering algorithms were used in the study: K-means and Fuzzy C-means. The trained clustering algorithms were used to group the training and the validation data of the segmentation model in [3] into clusters. The performance of the proposed method was assessed using the segmentation model. It was trained and validated on a reference dataset first and on different sets of sequences selected with the proposed methods. The final results were finally compared and discussed.

Since the model trained on a dataset created using the Fuzzy C-means clustering showed the best results, it can be further studied. Also, training for more epochs could be tried to see if the performance of any of the created models could be improved.

References

[1] HAY, A. and IAIN, J. N. Recent advances in the understanding and management of oropharyngeal cancer. *F1000Research*. 30 August 2018. Vol. 7, F1000 Faculty Rev-1362. DOI 10.12688/f1000research.14416.1.

[2] Head and Neck Awareness Month: [e-source]. *The National Foundation for Cancer Research (NFCR)*. [Accessed 01.06. 2022]. Available from: https://www.nfcr.org/blog/head-and-neck-cancer-awareness-month.

[3] De BIASE, A., SIJTSEMA, N. M., van DIJK, L., LANGENDIJK, J. A. and van OOIJEN, P. Slice-by-slice deep learning aided oropharyngeal cancer segmentation with adaptive thresholding for spatial uncertainty on FDG PET and CT images. arXiv preprint arXiv:2207.01623 [eess.IV]. 04 July 2022. Available from: https://arxiv.org/abs/2207.01623.

[4] ELRAHMAN, Sh.M.Abd and AJITH A. A Review of Class Imbalance Problem. *Journal of Network and Innovative Computing*. 2013. Vol. 1. p. 332-340. *ISSN 2160-2174*.

[5] KRAWCZYK B. Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intellingence*. 2016. no. 5 (4), p. 221–232. DOI 10.1007/s13748-016-0094-0.

[6] ZHOU, Q., REN C. and QI, S. An Imbalanced R-STDP Learning Rule in Spiking Neural Networks for Medical Image Classification. *IEEE Access*. 29 December 2020. Vol. 8, p. 224162-224177, DOI 10.1109/ACCESS.2020.3044646G.

[7] REZAEI, M., YANG, H. and MEINEL, C. Recurrent generative adversarial network for learning imbalanced medical image semantic segmentation. *Multimed Tools Application*. 07 February 2019. no. 79, p. 15329–15348. DOI 10.1007/s11042-019-7305-1.

[8] REZAEI, M., YANG, H., HARMUTH, K. and MEINEL, C. Conditional Generative Adversarial Refinement Networks for Unbalanced Medical Image Semantic Segmentation. *IEEE Winter Conference on Applications of Computer Vision (WACV)*. 07-11 January 2019. p. 1836-1845, DOI 10.1109/WACV.2019.00200.

[9] LE, N, LE, T., YAMAZAKI, K., BUI, T., LUU, K. and SAVIDES, M. Offset Curves Loss for Imbalanced Problem in Medical Segmentation. *25th International Conference on Pattern Recognition (ICPR).* 10-15 January 2021. p. 9189-9195, DOI 10.1109/ICPR48806.2021.9411921.

[10] SONG, B., LI, S., SUNNY, S., GURUSHANTH, K., MENDONCA, P., MUKHIA, N., PATRICK, S., GURUDATH, S., RAGHAVAN, S., TSUSENNARO, I., LEIVON, S.T. et al. Classification of imbalanced oral cancer image data from high-risk population. *J. of Biomedical Optics*. 23 October 2021. no. 26 (10), 105001. DOI 10.1117/1.JBO.26.10.105001.

[11] PHAM, T.-C., DOUCET, A., LUONG, C. -M., TRAN, C.-T. and HOANG, V.-D. Improving Skin-Disease Classification Based on Customized Loss Function Combined With Balanced Mini-Batch Logic and Real-Time Image Augmentation. *IEEE Access*. 14 August 2020. Vol. 8, p. 150725-150737. DOI 10.1109/ACCESS.2020.3016653.

[12] TEH, K., ARMITAGE, P., TESFAYE, S., SELVARAJAH, D., WILKINSON, I.D. Imbalanced learning: Improving classification of diabetic neuropathy from magnetic resonance imaging. *PLOS ONE.* 15 December 2020. Available from: https://doi.org/10.1371/journal.pone.0243907.

[13] HAN, Zh., WEI, B., ZHENG, Yu., YIN, Yi. And LI, K.Breast Cancer Multi-classification from Histopathological Images with Structured Deep Learning Model. *Scientific reports.* 23 June 2017. Vol. 7, Article number 4172. DOI 10.1038/s41598-017-04075-z

[14] JAPKOWICZ, N., MYERS, C. and GLUCK, M. A novelty detection approach to classification. *IJCAI.* 20 August 1995. p. 518-523. Corpus ID: 18433487.

[15] GAO, L., Zhang, L., LIU, Ch. and WU Sh. Handling imbalanced medical image data: A deep-learning-based one-class classification approach. *Artificial intelligence in medicine*. 07 August 2020. Vol. 108, 101935. DOI 10.1016/j.artmed.2020.101935.

[16] NUGRAHA, W., MAULANA, S. and SASONGKO, A. Clustering Based Undersampling for Handling Class Imbalance in C4.5 Classification Algorithm. *Journal of Physics: Conference Series. International Conference on Advanced Information Scientific Development (ICAISD), West Java, Indonesia*. 6-7 August 2020. Vol. 1641, no. 012014. DOI 10.1088/1742-6596/1641/1/012014.

[17] Zhang, J., Chen, L. and Abid, F. Prediction of Breast Cancer from Imbalance Respect Using Cluster-Based Undersampling Method. *Journal of Healthcare Engineering*. 16 October 2019. no. 2019:7294582. DOI 10.1155/2019/7294582.

[18] WAHL, R.L. Principles and Practice of PET and PET/CT. Published November 25th 2008 by LWW. ISBN 0781779995.

[19] VAQUERO, J.J., KINAHAN, P. Positron Emission Tomography: Current Challenges and Opportunities for Technological Advances in Clinical and Preclinical Imaging Systems. *Annual Review of Biomedical Engineering.* December 2015. Vol. 17:385-414. DOI 10.1146/annurev-bioeng-071114-040723.

[20] EL-KHODARY, M., TABASHY, R., OMAR, W., MOUSA, A. and MOSTAFA, A., The role of PET/CT in the management of head and neck squamous cell carcinoma. *The Egyptian Journal of Radiology and Nuclear Medicine.* June 2011. Vol. 42, Issue 2, p. 157-167, ISSN 0378-603X, DOI 10.1016/j.ejrnm.2011.05.006.

[21] HARVEY, A., ZIESSMAN, JANIS, O'MALLEY, P. and JAMES, H. Thrall, *Nuclear Medicine* (Fourth Edition). W.B. Saunders. 2014. p. 227-264, ISBN 9780323082990.

[22] HAGOS, Y.B., MINH, V.H., KHAWALDEH, S., PERVAIZ, U. and ALEEF, T.A. Fast PET Scan Tumor Segmentation Using Superpixels, Principal Component Analysis and K-Means Clustering. *Methods Protoc.* 8 January 2018. no 1 (1), 7. DOI 10.3390/mps1010007.

[23] MAH, K. and CALDWELL, C. B., chapter 4 – Biological Target Volume, Editor(s): PAULINO, A.C. *PET-CT in Radiotherapy Treatment Planning.* Elsevier, 2008. p. 52-89. ISBN 9781416032243.

[24] Luisa Cutillo, in Encyclopedia of Bioinformatics and Computational Biology. Amsterdam, Netherlands; Cambridge, MA, United States: Elsevier, 2019. ISBN 9780128114322.

[25] XU, R. and WUNSCH, D. Survey of Clustering Algorithms. Neural Networks. *IEEE Transactions on Neural Networks*. June 2005. no. 16 (3), p. 645-678. DOI 10.1109/TNN.2005.845141.

[26] KINAHAN, P.E. and FLETCHER J.W. Positron emission tomography-computed tomography standardized uptake values in clinical practice and assessing response to therapy. *Seminars in ultrasound CT and MR.* December 2010. Vol. 31 (6), p. 496-505. DOI 10.1053/j.sult.2010.10.001.

[27] LEE, D.K., IN, J. and LEE, S. (2015). Standard deviation and standard error of the mean. *Korean journal of anesthesiology*. June 2015. no 68 (3), 220-3. DOI 10.4097/kjae.2015.68.3.220.

[28] COOK, G.J.R., SIDDIQUE, M., TAYLOR, B.P. et al. Radiomics in PET: principles and applications. *Clinical and Translation Imaging 2*. 03 June 2014. p. 269–276. DOI 10.1007/s40336-014-0064-0.

[29] BHAGAT, P.K., CHOUDHARY, P., SINGH, Kh. M. Chapter 13 – A comparative study for brain tumor detection in MRI images using texture

features. Editor(s): DEY, N., CHAKI, J., KUMAR, R. In Advances in ubiquitous sensing applications for healthcare, *Sensors for Health Monitoring*. Academic Press, 2019. Vol. 5, p. 259-287, ISBN 9780128193617, DOI 10.1016/B978-0-12-819361-7.00013-0.

[30] MEYER, H.-J., PURZ, S., SABRI, O., SUROV, A. Relationships between histogram analysis of ADC values and complex 18F-FDG-PET parameters in head and neck squamous cell carcinoma. *PLOS ONE.* 06 September 2018. Available from: https://doi.org/10.1371/journal.pone.0202897.

[31] *NIST/SEMATECH e-Handbook of Statistical Methods.* 1. Exploratory Data Analysis, 1.3. EDA Techniques, 1.3.5. Quantitative Techniques. [Accessed 07.05.2022]. Available from: https://doi.org/10.18434/M32189.

[32] SENARATHNA, S. and HEMAPALA, K.T.M.U. Optimized Adaptive Overcurrent Protection Using Hybridized Nature-Inspired Algorithm and Clustering in Microgrids. *Energies.* June 2020. no 13 (13), p. 3324. DOI 10.3390/en13133324.

[33] SCHOTT, M. K-Means Clustering Algorithm for Machine Learning . *Capital One Tech* [online]. 23 Aprile 2019. [Accessed 14.05.2022]. Available from: https://medium.com/capital-one-tech/k-means-clustering-algorithm-for-machine-learning-d1d7dc5de882.

[34] SALUNKHE, V. K-Means Clustering. *Capital One Tech* [online]. 28 July 2021. [Accessed 25.05.2022]. Available from: https://medium.com/@viveksalunkhe80/k-means-clustering-b8e4ca9a75bb.

[35] DABBURA, D. K-means Clustering: Algorithm, Applications, Evaluation Methods, and Drawbacks. *Towards Data Science*. [Accessed 25.05.2022]. Available from: https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a.

[36] LI, K., LIU, Y., WANG, Q., WU, Ya., SONG, Sh., SUN, Y., LIU, T., WANG, J., LI, Y., DU, Sh. A Spacecraft Electrical Characteristics Multi-Label Classification Method Based on Off-Line FCM Clustering and On-Line WPSVM. *PLOS ONE*. 06 November 2015. Available from: https://journals.plos.org/plosone/article/authors?id=10.1371/journal.pone.0140395.

[37] MEENA, A. and RAJA, K. Segmentation of alzheimers disease in PET scan datasets using MATLAB. *International Journal on Information Sciences and Computing*, *arXiv preprint arXiv:1302.6426*. July 2012. Vol. 6, no 2, p. 44-48. Available from: https://arxiv.org/abs/1302.6426.

[38] Alifa98. Fuzzy-C-Means-Clustering. *GitHub*. 27 Augest 2021. [Accessed 07.06.2022]. Available from: https://github.com/alifa98/Fuzzy-C-Means-Clustering.

[39] VIRMANI, D., TANEJA, Sh., MALHOTRA, G. Normalization based K means Clustering Algorithm. *arXiv:1503.00900 [cs.LG]*. 03 Mach 2015. [Accessed 01.07.2022]. Available from: https://arxiv.org/abs/1503.00900.

[40] DE RIDDER,M, BALM, AJM, dE JONG, RJB, TERHAARD, CHJ, TAKES, RP, SLINGERLAND, M, DIK, E, SEDEE, RJE, DE VISSCHER, JGAM, BOUMAN, H, WILLEMS, SM, WOUTERS, MW, Smeele, LE, VAN DIJK, BAC. Variation in head and neck cancer care in the Netherlands A retrospective cohort evaluation of incidence, treatment and outcome. *European Journal of Surgical Oncology*, vol. 43, no. 8, p. 1494-1502. August 2017. Available from: https://doi.org/10.1016/j.ejso.2017.02.017

[41] EL EMAM, K, RODGERS, S, MALIN, B. Anonymising and sharing individual patient data. *Biomedical Journal*. bPMID: 25794882; PMCID: PMC4707567.. 20 March 2015. Awailable from: doi: 10.1136/bmj.h1139.

List of Appendices

Appendix 1: Comparison table of the model performance evaluation metrics

	Model 1 (baseline)			Model 2			Model 3			Model 4		
threshold	dice	precision	recall	dice	precision	recall	dice	precision	recall	dice	precision	recall
0.1	0.505	0.420	0.998	0.613	0.602	0.998	0.526	0.453	0.998	0.547	0.473	0.999
0.2	0.506	0.421	0.996	0.613	0.603	0.995	0.527	0.454	0.995	0.548	0.474	0.996
0.3	0.506	0.423	0.993	0.614	0.605	0.992	0.527	0.455	0.992	0.548	0.476	0.994
0.4	0.516	0.438	0.976	**0.617**	**0.616**	**0.980**	0.535	0.469	0.978	0.557	0.491	0.982
0.5	0.516	0.439	0.973	0.617	0.617	0.977	0.536	0.471	0.976	0.558	0.492	0.980
0.6	0.517	0.441	0.971	0.617	0.618	0.975	0.536	0.472	0.973	0.558	0.493	0.978
0.7	0.526	0.457	0.952	0.619	0.627	0.964	0.542	0.485	0.959	0.563	0.504	0.967
0.8	0.526	0.458	0.949	0.619	0.628	0.961	0.543	0.486	0.956	0.564	0.505	0.964
0.9	0.527	0.459	0.947	**0.619**	**0.629**	**0.959**	0.543	0.487	0.954	0.564	0.506	0.962

	Model 5			Model 6			Model 7			Model 8		
threshold	dice	precision	recall	dice	precision	recall	dice	precision	recall	dice	precision	recall
0.1	0.556	0.483	0.997	0.573	0.511	0.998	0.583	0.547	0.998	0.603	0.572	0.998
0.2	0.557	0.485	0.994	0.574	0.512	0.996	0.583	0.548	0.996	0.604	0.574	0.995
0.3	0.557	0.487	0.990	0.574	0.514	0.993	0.584	0.549	0.994	0.604	0.576	0.991
0.4	0.565	0.501	0.978	0.580	0.524	0.983	0.588	0.558	0.986	0.610	0.590	0.971
0.5	0.566	0.503	0.975	0.580	0.526	0.981	0.588	0.559	0.984	**0.610**	**0.592**	**0.969**
0.6	0.567	0.505	0.972	0.581	0.528	0.978	0.589	0.560	0.983	0.611	0.594	0.966
0.7	0.577	0.526	0.951	0.587	0.542	0.959	0.591	0.566	0.977	0.615	0.605	0.954
0.8	0.577	0.528	0.948	0.587	0.543	0.956	0.592	0.567	0.975	0.616	0.607	0.952
0.9	0.577	0.529	0.945	0.587	0.544	0.954	0.592	0.568	0.973	**0.616**	**0.608**	**0.949**

Appendix 2: Python code. Features extraction and creating clusters with the help of K-Means and Fuzzy-C means algorithms

```
!pip install monai==0.7.0 --user
!pip install "git+https://github.com/Project-
MONAI/MONAI#egg=monai[nibabel,ignite,tqdm]" --user
!pip install matplotlib --user
!pip install --user scikit-learn
!pip install --user seaborn
!pip install --user fuzzy-c-means


import glob
import argparse
import random
import os
import random
import json
import sys

sys.dont_write_bytecode = True
import random
from PIL import Image
import matplotlib.pyplot as plt
from matplotlib import cm
import tempfile
import nibabel as nib
import SimpleITK as sitk
import numpy as np
import pandas as pd

from scipy.stats import kurtosis
from scipy.stats import skew

import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.model_selection import train_test_split

import monai
from monai.config import print_config
from monai.utils import first
from monai.config import KeysCollection
from monai.data import Dataset, ArrayDataset, create_test_image_3d,
 DataLoader, CacheDataset
from monai.transforms import (
    Transform,
    MapTransform,
```

```
    Randomizable,
    AddChannel,
    AddChanneld,
    Compose,
    LoadImage,
    LoadImaged,
    Lambda,
    Lambdad,
    RandSpatialCrop,
    RandSpatialCropd,
    ThresholdIntensityd,
    NormalizeIntensityd,
    ScaleIntensityd,
    ConcatItemsd,
    RandFlipd,
    ToTensor,
    ToTensord,
    Orientation,
    Rotate,
    Resize,
    CenterSpatialCrop,
)
print_config()


class Resized_mine(MapTransform):

    def __init__(self, keys, spatial_size=(144,144,144)):

        super().__init__(keys)
        self.spatial_size = spatial_size
        self.resizer = CenterSpatialCrop(self.spatial_size)

    def resize(self, img):

        if np.shape(img)!=self.spatial_size:
            return img[:144,:144,:144] #self.resizer(img)
        else:
            return img


    def __call__(self, dictionary):

        dictionary = dict(dictionary)

        for key in self.keys:
            dictionary[key] = self.resize(dictionary[key])

        return dictionary
```

```
def dataset_prep(lists):
    dataset_train=[]
    for lista in lists:
        dataset_train.append(
        {'ct':lista[1],
        'pet':lista[2],
        'gtv':lista[3],
        'slice':lista[4],
        'ID':lista[0]
    })
    return dataset_train

class Create_sequences(MapTransform):

    def __init__(self, keys, seq = 3, plane = "x"):

        super().__init__(keys)
        self.sequences = seq
        self.plane=plane
        #self.modality=modality

    def slicing(self, img, starting):

        indx=starting
        sequence=[]

        for i in range(indx, indx+self.sequences):

            if self.plane=="x":
                ima = np.rot90(img[:,:,i],3)
            elif self.plane=="y":
                ima = np.rot90(img[:,i,:])
            elif self.plane=="z":
                ima = np.rot90(img[i,:,:])

            sequence.append(ima)

        return np.hstack(sequence)

    def __call__(self, dictionary):

        dictionary = dict(dictionary)
        starting = dictionary["slice"]

        for key in self.keys:
            dictionary[key] = self.slicing(dictionary[key],starting
) #self.create_sequence.
```

```
        return dictionary

def initialize_transform(images_keys, plane_, ct_norm="z_norm", pet
_norm="z_norm", test=False):

    normalize_ct={
        "z_norm": NormalizeIntensityd(("ct"), subtrahend=None, divi
sor=None, nonzero=False, channel_wise=False, allow_missing_keys=Fal
se),
        "min_max_norm": NormalizeIntensityd(("ct"), subtrahend=None
, divisor=None, nonzero=False, channel_wise=False, allow_missing_ke
ys=False)
    }
    normalize_pet={
        "z_norm": NormalizeIntensityd(("pet"), subtrahend=None, div
isor=None, nonzero=False, channel_wise=False, allow_missing_keys=Fa
lse),
        "min_max_norm": NormalizeIntensityd(("pet"), subtrahend=Non
e, divisor=None, nonzero=False, channel_wise=False, allow_missing_k
eys=False)
    }

    load_seq=[
        LoadImaged(keys=images_keys),
        Resized_mine(keys=images_keys, spatial_size=(144,144,144)),
    ]

    #if test==False:
        #load_seq.append(RandFlipd(keys=images_keys, prob=0.5, spat
ial_axis=2))

    pre_processing_ct=[
        ThresholdIntensityd(("ct"), threshold=-
1024, above=True, cval=-1024, allow_missing_keys=False),
        ThresholdIntensityd(("ct"), threshold=1024, above=False, cv
al=1024, allow_missing_keys=False),
    ]

    pre_processing_pet=[
        ThresholdIntensityd(("pet"), threshold=0, above=True, cval=
0.0, allow_missing_keys=False)
    ]

    try:
        pre_processing_ct.append(normalize_ct[ct_norm])
    except KeyError:
        "CT normalization technique not available!"

    try:
```

```
            pre_processing_pet.append(normalize_pet[pet_norm])
        except KeyError:
            "PET normalization technique not available!"

        sequence_prep=[
            Create_sequences(keys=images_keys, plane=plane_),
            AddChanneld(keys=images_keys),
            ConcatItemsd(keys=["ct","pet"], name="input"),
            ToTensord(keys=["input", "gtv"])
        ]

        train_tranforms = Compose(load_seq+pre_processing_ct+sequence_p
rep) #pre_processing_pet

        return train_tranforms

def initialize_transform_norm(images_keys, plane_, ct_norm="z_norm"
, pet_norm="min_max_norm", test=False):


        normalize_pet={
            "z_norm": NormalizeIntensityd(("pet"), subtrahend=None, div
isor=None, nonzero=False, channel_wise=False, allow_missing_keys=Fa
lse),
            "min_max_norm": ScaleIntensityd(("pet"), minv=0.0, maxv=1.0
, allow_missing_keys=False)
        }

        load_seq=[
            LoadImaged(keys=images_keys),
            Resized_mine(keys=images_keys, spatial_size=(144,144,144)),
        ]

        #if test==False:
            #load_seq.append(RandFlipd(keys=images_keys, prob=0.5, spat
ial_axis=2))

        pre_processing_ct=[
            ThresholdIntensityd(("ct"), threshold=-
1024, above=True, cval=-1024, allow_missing_keys=False),
            ThresholdIntensityd(("ct"), threshold=1024, above=False, cv
al=1024, allow_missing_keys=False),
        ]

        pre_processing_pet=[
            ThresholdIntensityd(("pet"), threshold=0, above=True, cval=
0.0, allow_missing_keys=False)
        ]
```

```
    # try:
        # pre_processing_ct.append(normalize_ct[ct_norm])
    # except KeyError:
       #  "CT normalization technique not available!"

     try:
         print('Normalization added')
         pre_processing_pet.append(normalize_pet[pet_norm])
     except KeyError:
         "PET normalization technique not available!"

     sequence_prep=[
         Create_sequences(keys=images_keys, plane=plane_),
         AddChanneld(keys=images_keys),
         ConcatItemsd(keys=["ct","pet"], name="input"),
         ToTensord(keys=["input", "gtv"])
     ]

     train_tranforms = Compose(load_seq+pre_processing_pet+sequence_
prep) #pre_processing_pet

     return train_tranforms



def read_split_file(path):

     with open(path, 'r') as inf:
         dict_from_file = eval(inf.read())

     return dict_from_file

#### Main

split_data = read_split_file('/data/s4880641/model_test/1/image_spl
it_new.json')
random.seed(100)


train_list = dataset_prep(split_data['training'])
validate_list = dataset_prep(split_data['validate'])

image_keys=('ct', 'pet', 'gtv')

train_tranforms_m = initialize_transform(image_keys, 'x', ct_norm="
z_norm", pet_norm="z_norm", test=False)

train_ds_m = Dataset(data=train_list, transform=train_tranforms_m)
```

```
train_loader_m = DataLoader(train_ds_m, batch_size=1, shuffle=True,
 num_workers=0, pin_memory=True)
im_train_m=(train_ds[5]["pet"])
im_train_id =(train_ds[5]["ID"])
im_train_slice =(train_ds[5]["slice"])


%matplotlib inline
plt.imshow(im_train[0])



def plot_hist(pet, name):

    plt.hist(pet, bins=15)
    plt.show()
    #plt.savefig('/data/s4880641/histogram/'+name+'_hist.png')

def entropy(signal):
        '''
        function returns entropy of a signal
        signal must be a 1-D numpy array
        '''
        lensig=signal.size
        symset=list(set(signal))
        numsym=len(symset)
        propab=[np.size(signal[signal==i])/(1.0*lensig) for i in sy
mset]
        ent=np.sum([p*np.log2(1.0/p) for p in propab])
        return ent

def en(im_arr):
    #im_arr = im_train[0]
    im = Image.fromarray(np.uint8(cm.plasma(im_arr)*255)) #entropy
    #plt.imshow(im)
    greyIm=im.convert('L')
    greyIm = np.array(greyIm)

    N=5
    S=greyIm.shape
    E=np.array(greyIm)
    for row in range(S[0]):
            for col in range(S[1]):
                    Lx=np.max([0,col-N])
                    Ux=np.min([S[1],col+N])
                    Ly=np.max([0,row-N])
                    Uy=np.min([S[0],row+N])
                    region=greyIm[Ly:Uy,Lx:Ux].flatten()
                    E[row,col]=entropy(region)
```

```
    return np.mean(E)


im_arr = im_train[0]
im = Image.fromarray(np.uint8(cm.plasma(im_arr)*255)) #entropy
#plt.imshow(im)
greyIm=im.convert('L')
greyIm = np.array(greyIm)

N=5
S=greyIm.shape
E=np.array(greyIm)
for row in range(S[0]):
        for col in range(S[1]):
                Lx=np.max([0,col-N])
                Ux=np.min([S[1],col+N])
                Ly=np.max([0,row-N])
                Uy=np.min([S[0],row+N])
                region=greyIm[Ly:Uy,Lx:Ux].flatten()
                E[row,col]=entropy(region)

%matplotlib inline
plt.figure(figsize=(20,10))
plt.subplot(1,3,1)
plt.imshow(im)

plt.subplot(1,3,2)
plt.imshow(greyIm, cmap=plt.cm.gray)

plt.subplot(1,3,3)
plt.imshow(E, cmap=plt.cm.jet)
plt.xlabel('Entropy') # in 10x10 neighbourhood

plt.colorbar(fraction=0.046, pad=0.08)

plt.show()


##Creating Dataframe with extracted features

i=0
df = pd.DataFrame(columns = ["ID", "slice", "meanSUV", "maxSUV", "m
inSUV", "std", "skewness", "kurtosis", "entropy"])

for image in train_ds_m:
    #i=i+1
    pet = image["pet"][0]
    id_ = image["ID"]
    slice_ = image["slice"]
```

```
    maxv = np.max(pet)
    mean = np.mean(pet)

    minv = np.min(pet)
    std = np.std(pet)
    sk = np.mean(skew(pet))
    kurt = np.mean(kurtosis(pet))
    entropy_mean = en(pet)

    df= df.append({'ID': id_, 'slice': slice_, 'maxSUV': maxv, 'min
SUV': minv, 'meanSUV': mean, 'std': std, 'skewness': sk, 'kurtosis'
:kurt, 'entropy':entropy_mean}, ignore_index=True)

df.to_csv('/data/s4880641/dataframe/table_f.csv', index = False, he
ader=True)


#Data representation

df = pd.read_csv('/data/s4880641/dataframe/table_final.csv')
df_analysis=df.drop(['ID','slice'], axis=1)
df_analysis.to_csv('/data/s4880641/dataframe/table_work.csv', index
 = False, header=True)
df_analysis

# Visualizing the correlation of the data and identifying variables
 for further analysis
g = sns.PairGrid(df_analysis, hue = "tumor_no tumor")
g.map(sns.scatterplot)
g.add_legend()

#K-Means clustering


df_fin=df_analysis.dropna()
data =df_fin[~df_fin.isin([np.nan, np.inf, -np.inf]).any(1)]
data.to_csv('/data/s4880641/dataframe/data.csv', index = False, hea
der=True)

X = pd.DataFrame(data).to_numpy()

# Determine optimal cluster number with elbow method
wcss = []

for i in range(1, 11):
    model = KMeans(n_clusters = i,
                   init = 'k-means++',
                   max_iter = 300,
                   n_init = 10,
```

```
                    random_state = 0)
    model.fit(data)
    wcss.append(model.inertia_)

# Show Elbow plot
plt.plot(range(1, 11), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('Within Cluster Sum of Squares (WCSS)')

boolMarker = data["tumor_no tumor"].to_numpy()

#Model for k-means clustering predicting
kmeans = KMeans(n_clusters=3, init="k-
means++", max_iter=50000, n_init=10, random_state=0)
pred_y = kmeans.fit_predict(data)

#Visualization of K-means clustering using two features

def split(arr,colorArr,cond):
    return [arr[cond], arr[~cond], colorArr[cond], colorArr[~cond]]

[data1, data2, color1, color2] = split(X,kmeans.labels_,boolMarker=
=1)
%matplotlib inline
plt.scatter(data1[:,1],data1[:,6],c=color1, marker="+", alpha=0.4)
plt.scatter(data2[:,1],data2[:,6],c=color2,marker="d", alpha=0.7)
plt.scatter(kmeans.cluster_centers_[:,1],kmeans.cluster_centers_[:,
6],s=100,c='red',marker="+")
plt.title("K means clustering") #title
plt.xlabel("Maximum SUV") #x label
plt.ylabel("entropy") #y label
plt.legend(["Tumor" , "NoTumor"], loc= 'upper right')
plt.show()


kmeans.cluster_centers_[:,0]
kmeans.cluster_centers_[:,1]
kmeans.inertia_
sns.countplot(pred_y)


#FUZZY C MEANS CLUSTERING

from fcmeans import FCM

fcm = FCM(n_clusters=3)
fcm.fit(X)
```

```
# outputs
fcm_centers = fcm.centers
fcm_labels = fcm.predict(X)

# plot result
boolMarker = data["tumor_no tumor"].to_numpy()

def split(arr,colorArr,cond):
    return [arr[cond], arr[~cond], colorArr[cond], colorArr[~cond]]

[data1, data2, color1, color2] = split(X,fcm_labels,boolMarker==1)
%matplotlib inline
plt.scatter(data1[:,0],data1[:,1],c=color1, marker="+", alpha=0.4)
plt.scatter(data2[:,0],data2[:,1],c=color2,marker="d", alpha=0.7)
plt.scatter(fcm_centers[:,0],fcm_centers[:,1],s=100,c='red',marker=
"+")
plt.title("Fuzzy-c means means clustering") #title
plt.xlabel("clusters") #x label
plt.ylabel("Maximum SUV") #y label
plt.legend(["Tumor" , "NoTumor"], loc= 'upper right')
plt.show()
```

Appendix 3: Python code. Sequence selection algorithm. Training and validation datasets (with the help of K-means clustering according to all features)

```
import random
random.seed(100)


dfa = pd.read_csv('/data/s4880641/split/training_data.csv')
dfa=dfa.dropna()
data = pd.read_csv('/data/s4880641/dataframe/data.csv')


X = pd.DataFrame(data).to_numpy()
Y=whiten(X)
model = KMeans(n_clusters=3, init="k-
means++", max_iter=50000, n_init=10, random_state=0)
model.fit(Y)

dat=dfa.drop(['ID','slice'], axis=1)

y=whiten(dat)
pred_y = model.fit_predict(y)

model.labels_
dfa['cluster']=model.labels_
dfa.to_csv('/data/s4880641/dataframe/table_clust.csv', index = Fals
e, header=True)

dfc = pd.read_csv('/data/s4880641/dataframe/table_clust3.csv')

%matplotlib inline
sns.countplot(pred_y)

#Identifying amount of tumor/no tumor sequences in the dataset and
calculating 20% of the dataset
dfa['tumor'].value_counts()
n= 1561/4

#Clusters are assigned to the data

df1_tumor=dfc[dfc['tumor']==1]
df1_notumor=dfc[dfc['tumor']==0]

df0 = df1_notumor[df1_notumor['cluster']==0]
df1 = df1_notumor[df1_notumor['cluster']==1]
df2 = df1_notumor[df1_notumor['cluster']==2]
```

```
#Selecting equal amount of sequences from each cluster within 20%

d0 = df0.sample(n=130)
d1 = df1.sample(n=130)
d2 = df2.sample(n=130)

#Final training dataframe
pieces=[df1_tumor,d0,d1,d2]
result = pd.concat(pieces)
result=result.drop(['meanSUV','maxSUV','minSUV','std', 'skewness','
kurtosis','entropy','tumor','cluster' ], axis=1)
result.to_csv('/data/s4880641/dataframe/table_training_all.csv', in
dex = False, header=True)
result


dfa = pd.read_csv('/data/s4880641/split/validate_data.csv')
dfa=dfa.dropna()
data = pd.read_csv('/data/s4880641/dataframe/data.csv')

import random
random.seed(100)

X = pd.DataFrame(data).to_numpy()
Y=whiten(X)
model = KMeans(n_clusters=3, init="k-
means++", max_iter=50000, n_init=10, random_state=0)
model.fit(Y)

dat=dfa.drop(['ID','slice'], axis=1)

y=whiten(dat)
pred_y = model.fit_predict(y)

model.labels_
dfa['cluster']=model.labels_
dfa.to_csv('/data/s4880641/dataframe/table_clust.csv', index = Fals
e, header=True)

dfc = pd.read_csv('/data/s4880641/dataframe/table_clust3.csv')

%matplotlib inline
sns.countplot(pred_y)

#Identifying amount of tumor/no tumor sequences in the dataset and
calculating 20% of the dataset
dfa['tumor'].value_counts()
n= 702/4
```

```
#Clusters are assigned to the data

df1_tumor=dfc[dfc['tumor']==1]
df1_notumor=dfc[dfc['tumor']==0]

df0 = df1_notumor[df1_notumor['cluster']==0]
df1 = df1_notumor[df1_notumor['cluster']==1]
df2 = df1_notumor[df1_notumor['cluster']==2]

#Selecting equal amount of sequences from each cluster within 20%

d0 = df0.sample(n=59)
d1 = df1.sample(n=58)
d2 = df2.sample(n=59)

#Final training dataframe
pieces=[df1_tumor,d0,d1,d2]
result = pd.concat(pieces)
result=result.drop(['meanSUV','maxSUV','minSUV','std', 'skewness','
kurtosis','entropy','tumor','cluster' ], axis=1)
result.to_csv('/data/s4880641/dataframe/table_validate_all.csv', in
dex = False, header=True)
result
```